# RoboMed Software Design

**CSCI-P465/565 (Software Engineering I)**

**Project Team Aravind Ramalingam Daksha Nagre Eugene Wang**



1. **Introduction**
   1. **System Description**

Create a platform to facilitate interactions between patients, doctors, and insurance providers. In a patient-doctor relationship, a patient can look up doctors, read their reviews, and schedule appointments; a doctor can gain exposure to platform’s patient users and when an appointment is scheduled, a doctor can access a patient’s medical records, insurance records, and bill the patient through the system. In a patient-insurance provider relationship, a patient can compare and buy insurance plans that insurance providers’ offers. This platform increases efficiency in the medical insurance market by putting insurers and patients in one platform hence reducing the time consumers spend on researching and comparing insurance providers and plans. This platform also simplifies the process of a patient doctor interactions by putting searching, scheduling, and billing process all in one place.

# Design Evolution

* + 1. **Design Issues**
* No inherent issues with the technologies or designed components.
* An outside issue is access to web hosting solutions

# Candidate Design Solutions

* MEAN stack
  + Abbreviation for MongoDB, Express, Angular, and Node.js
  + It’s a technology bundle that covers solutions needed from server side to the client side.
  + MongoDB is a NoSQL database system. Express is a framework used to build web apps in Node. Angular is a js library for front end development. Node is a server side javascript execution environment
* MERN stack
  + Abbreviation for MongoDB, Express, React, and Node.js
  + It’s a technology bundle that covers solutions needed from server side to the client side.
  + MongoDB is a NoSQL database system. Express is a framework used to build

web apps in Node. React is a js framework solution for front end development. Node is a server side javascript execution environment

# Design Solution Rationale

* We chose the MERN stack over MEAN stack.
* The only difference between them is the framework used for front end development
* React used virtual DOM which makes it a lot faster than Angular. In the event of dealing with hundreds or thousands of data requests, using React would give a faster unser experience.
* Team members less familiar with ReactJS, but given the popularity and abundance of online resources we believe we could learn it fast enough to use it.

# Design Approach

* + 1. **Methods**

We are implementing Command Design Pattern technique. This design pattern enables us to decouple requester of a particular action from the object that performs the action.

The Command Design Pattern makes our code extensible, so we can add new commands in our system without making changes in the existing code.

So, the methods which are implemented will be designed in such a way that the requester methods does not know request is getting fulfilled.

# Standards

Safety standards:

* + - * Pushing new code into master branch of git repo must be approved by at least two members of the team
      * Otherwise, new changes should be pushed to a separate branch Design Plan Standards:
      * Before actual implementation of features, discussion between team members should occur. What the user experience (front end) would be designed should be agreed with the team. The backend implementation can be decided by the person responsible for it, then later communicate to other members if need be.

# Tools

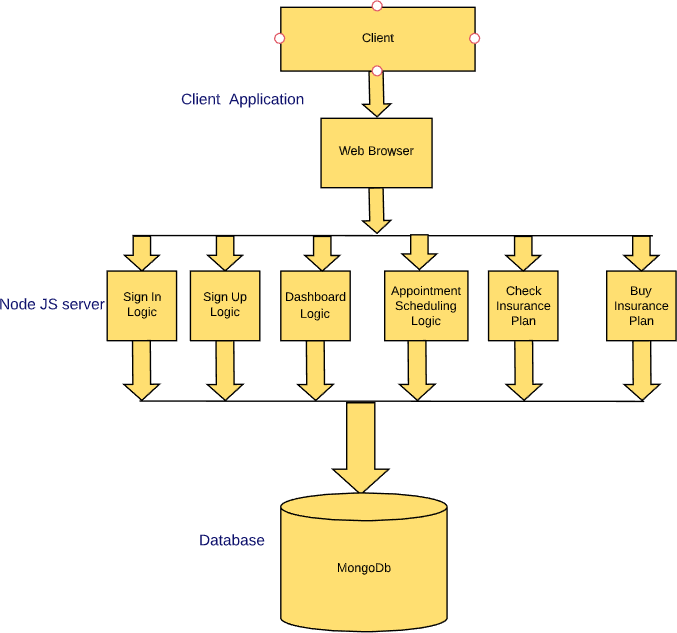
We are using following tools in our project:

* **MongoDB:** It is a NOSQL database. One of the perks of using Mongo is data is stored in the form of JSON style documents and we don’t have to write complex and expensive joins. Using MongoDB we will create our database.
* **React:** React is a javascript library used for building user interfaces. We are using React for designing simple views for each state in our application. React will help us to generate single page applications.
* **Express:** Express is a Node JS framework. Express framework facilitates us in following ways:
  + We are using it to set up middlewares to respond to HTTP Requests.
  + Define Routing tables.
  + Dynamically render HTML pages based on passing arguments to templates.
* **Node:** Enables easy scalability, Multithreaded server, High performance and advanced caching.

# System Architecture

* 1. **System Design**

The client machine uses a web browser to view and interact with a web application front end designed with React. The client interacts with the backend server implemented with nodeJS and controlled with Express. The backend server interacts with a database implemented in MongoDB



# External Interfaces

External interfaces include Google reCaptcha and Duo authentication for Two factor.

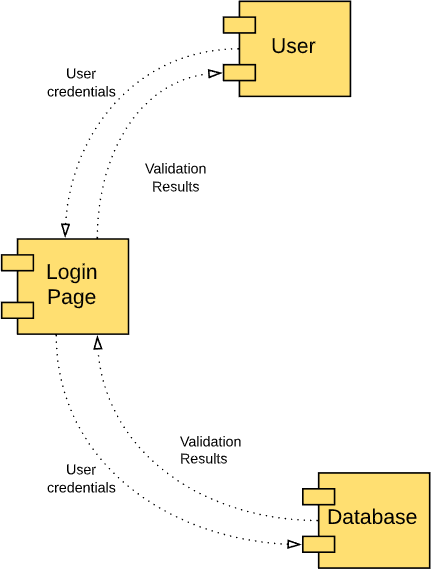
# Component Design

The components that we have implemented in our project so far are as follows:

**1.**

* **Component Name:** Sign In
* **Component Description:** Using this component, our users can request access to their dashboard page.
* **Responsible Development Team Member:** Daksha and Aravind

# Component Diagram:



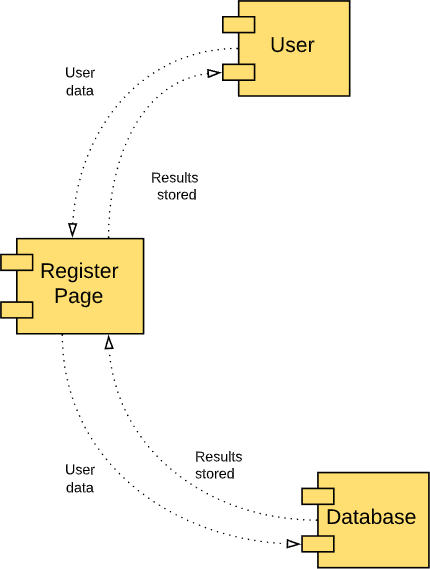
* **Component User Interface:** The user will see two forms corresponding to the email and password in the sign in screen and a submit button.
* **Component Objects:** Methods to search and locate user profile data will be used
* **Component Interfaces (internal and external):** On submission of email and password, that data gets sent back to the server. If the pair of email and password matches one of the user profile on record, user will be redirected to the tfa page for next step verification.
* **Component Error Handling:** Must be valid username and password. Password must be decrypted accurately.

**2.**

* **Component Name:** Register
* **Component Description:** Using this component, our users can sign up for an account

# Responsible Development Team Member: Daksha

* **Component Diagram:**

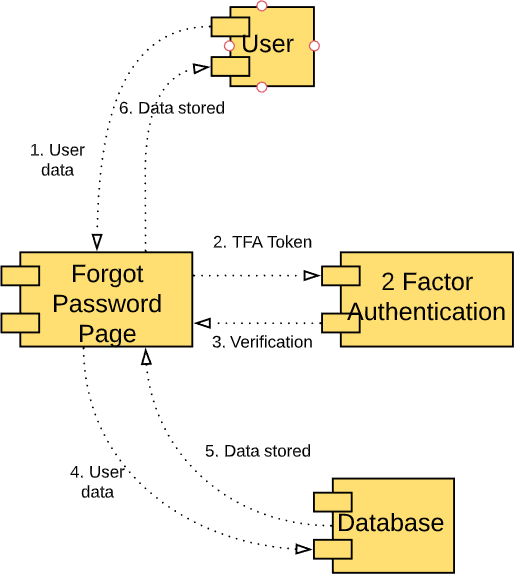


* **Component User Interface:** The user will see two forms corresponding to the email and password and a submit button.
* **Component Objects:** Methods to create and insert user profile data will be used
* **Component Interfaces (internal and external):** On submission of email and password, that data gets sent back to the server. If the pair of email hasn’t been used before, the email and password will be used to create a new user profile on record, user will be redirected to a page to set up tfa.
* **Component Error Handling:** The email must be unique. Users are to be validated before signup.

**3.**

* **Component Name:** Forget password
* **Component Description:** Using this component, our users can reset their password

# Responsible Development Team Member: Aravind

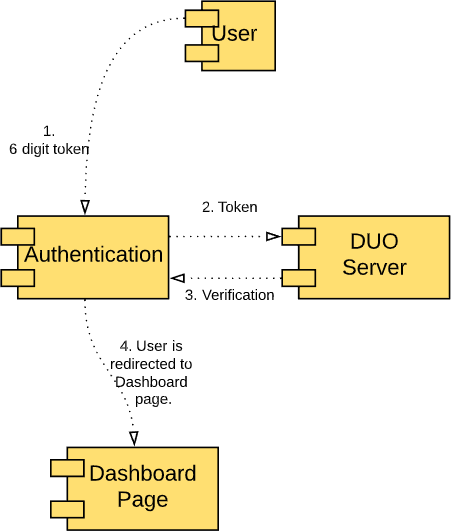
* **Component Diagram:**
* **Component User Interface:** The user will see a form prompting them to enter their registered email address and a submit button. Then user will see a page with a form prompting them to enter their tfa token. After that they will see two forms that lets then enter their password to reset it.
* **Component Objects:** Methods to search and locate user profile, calculate tfa token value, and create and insert user password data.
* **Component Interfaces (internal and external):** On submission of the email, that data gets sent back to the server. If there’s a user with the email, we prompt them with tfa verification screen. The tfa verification api will be called, they user pass it, they get to send us a new password, and with that we will overwrite the old password in our db.
* **Component Error Handling:** It requires further analysis.

# 3.

* **Component Name:** Two Factor Authentication
* **Component Description:** Two factor authentication for login and reset password.

# Responsible Development Team Member: Eugene

* **Component Diagram:**

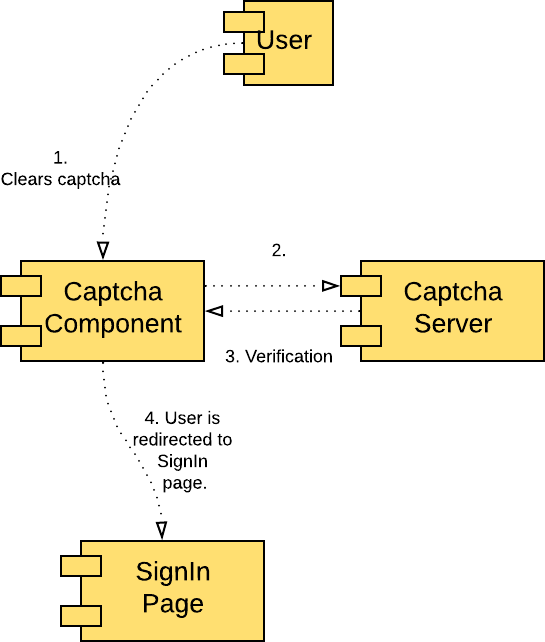


* **Component User Interface:** User will be prompted to enter a 6 digit token in a form, and submit it by clicking a submit button.
* **Component Objects:** Methods for token calculation, random private key generation, qr code conversion, and pulling users secret key will be used
* **Component Interfaces (internal and external):** During tfa setup, a secret key is generated in the server and converted to qr code to be displayed to the user; the user then scan the qr code(secret key) into their phone’s tfa app, completing the setup. During verification, the 6 digit token is calculated by the secret key and the current timestamp on both user and us, if the user’s token matches ours, users will be verified.
* **Component Error Handling:** It requires further analysis.

# 4.

* **Component Name:** Captcha
* **Component Description:** Captcha trigger when user failed three login attempts
* **Responsible Development Team Member:** Aravind, Eugene

# Component Diagram:



* **Component User Interface:** At the third failed attempt in signing in. A captcha will appear on the same page prompting user to complete it before he can make further sign in attempts.
* **Component Objects:** Methods will be used for counting and resetting the number of failed login attempts, deny user further login attempts, sending a captcha request, receiving captcha result, lifting the ban on login attempt.
* **Component Interfaces (internal and external):** We keep track of a user’s failed login attempts, and when it reaches three, we send them a google’s captcha challenge embedded with our website’s identification key to complete. When the user completes and submits the challenge to google, we will request the result of the challenge with the private key. If the user passed it we reset the login attempt counter, if not we send the user another captcha.
* **Component Error Handling:** It requires further analysis.

# 5.

* **Component Name:** Scheduler
* **Component Description:** Scheduler for doctor to set appointment times and accept appointment requests. And for user to send appointment requests.

# Responsible Development Team Member: Aravind

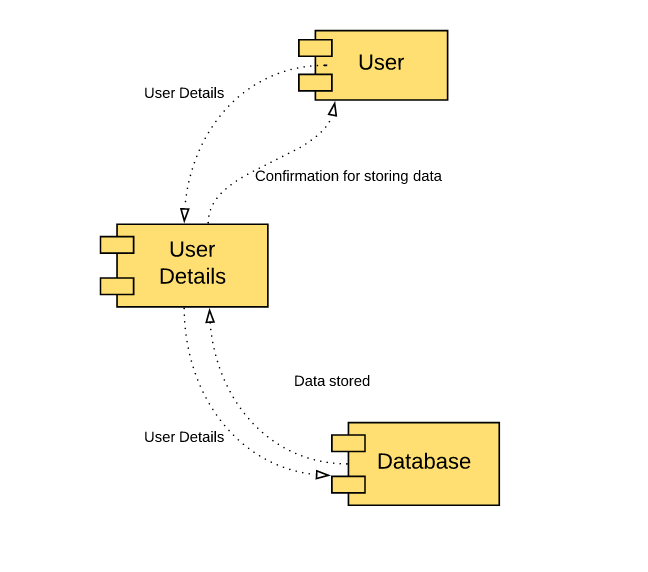
* **Component Diagram: n/a**
* **Component User Interface:** Doctor will be able to enter their hours for the week, effectively setting their available appointment times; see patient’s appointment requests and accept or deny them. Patient will be able to fill out an appointment request and send it to the specified doctor.
* **Component Objects:** Methods to retrieve and insert schedule data and appointment request data will be used.
* **Component Interfaces (internal and external):** On setting up schedule for doctor users, a form will be filled out with the available hours and stored in the DB to be requested by profile page or patient users. On accepting or denying patient user’s appointment request, the request status in the db will be updated; and the user will be able to see the updated status. When patient users want to schedule an appointment, a form containing the doctor’s name, appointment time, and comments will be filled out and submitted. On submission, we created a new entry for the appointment request containing patient id, doctor id, appointment id, time, and comments.
* **Component Error Handling:** It requires further analysis.

# 6.

* **Component Name:** User Details
* **Component Description:** Using this component, user can edit their profile details.

# Responsible Development Team Member: Aravind, Daksha

* **Component Diagram:**

****

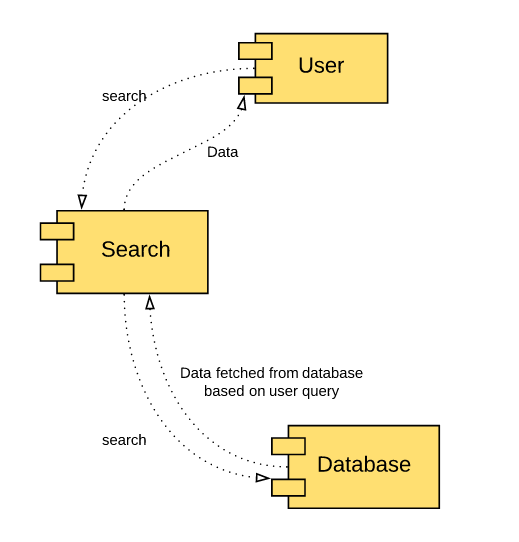
* **Component User Interface:** User gets to edit his or her profile page using this component. For each role there is a different view. We are doing role based content rendering. So, doctor has a different view. Patient has a different view and insurance agent has a different view.
* **Component Objects:** Methods to retrieve and insert user details data.
* **Component Interfaces (internal and external):** When user signs in, he or she gets the option to edit his or her profile page and upload a profile picture from here. When user clicks on the edit tab, a window will pop open which contains the user details. User edit only those data that he wishes to update. After submitting the form, we update the data for that particular user in the database.
* **Component Error Handling:** It requires further analysis.

7.

* **Component Name:** Search
* **Component Description:** Using this component, patients can search for doctor and insurance agent.

# Responsible Development Team Member: Aravind

* **Component Diagram:**

****

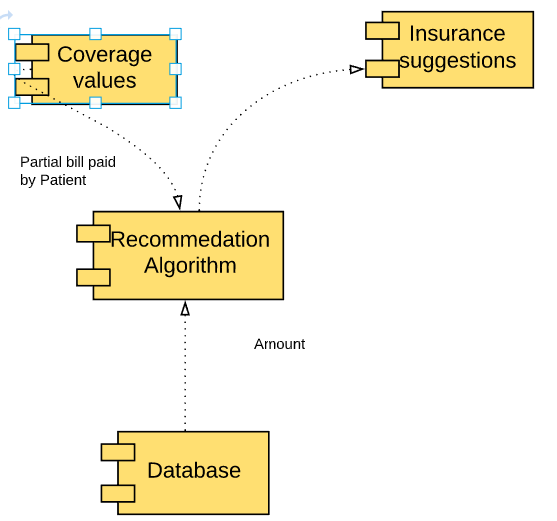
* **Component User Interface:** Whenever a user signs in, he can use the search feature provided on the dashboard. The search feature allows the patient to search doctor. Patient can search doctor by typing his name in the search box or by typing specialization keyword in the search box. Patient can also search for insurance agent by typing his name. But patient cannot search for another patient using the search feature. Doctor can use the search feature to search for another doctor or an insurance agent. Insurance agent can use the search feature to search doctor or another insurance agent. The search also has text auto-completion feature.
* **Component Objects:** Methods to retrieve data based on user’s search query.
* **Component Interfaces (internal and external):** When user signs in, he or she gets the option to edit his or her profile page. When user clicks on the edit tab, a window will pop open which contains the user details. User edit only those data that he wishes to update. After submitting the form, we update the data for that particular user in the database.
* **Component Error Handling:** It requires further analysis.

8.

* **Component Name:** Insurance Selection & Recommendation
* **Component Description:** Using this component, patients can purchase insurance from any insurance provider

# Responsible Development Team Member: Aravind, Eugene

* **Component Diagram:**

****

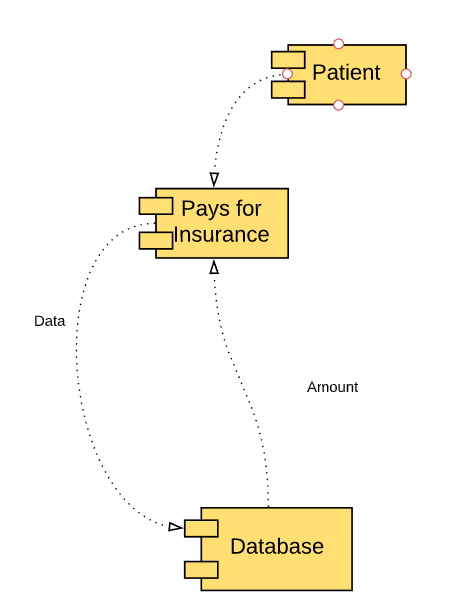
* **Component User Interface:** Whenever a user signs in, he can use the insurance feature provided on the dashboard. If there’s no insurance, there will be a prompt to ask him to purchase it. The purchase process involves the patient entering the desired max coverage amount, and he will be shown the recommended insurance providers calculated by our system. If a patient has insurance, he may cancel it.
* **Component Objects:** Forms to take numeric input, methods to do arithmetic for price calculations
* **Component Interfaces (internal and external):** When user signs in, he or she gets the option to buy or cancel his insurance package. When user without insurance clicks on the insurance tab, a purchase button will be available. On clicking purchase, a user will enter the desired coverage amount. And we will check the price of his patient’s desired plan for every provider we have in the database, and suggest to the patient the cheapest ones. Upon selecting purchase insurance, a new insurance object will be created under the user’s profile.
* **Component Error Handling:** It requires further analysis.

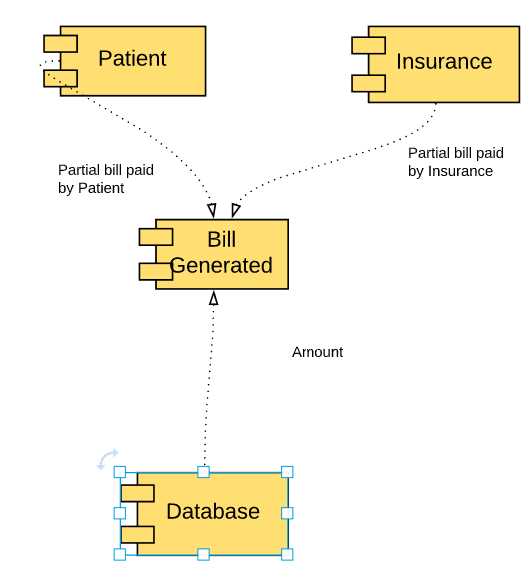
9.

* **Component Name:** Billing
* **Component Description:** Using this component, users can send bills to each other

# Responsible Development Team Member: Aravind

* **Component Diagram:**

****

****

* **Component User Interface:**

When a patient purchase an insurance plan, he will get a bill for the monthly amount of that insurance plan.

When a doctor completes an appointment report, the insurance provider gets billed for the full amount to the doctor, and the patient get billed the deductible to the insurance provider

* **Component Objects:** Bill object, consisting of party id’s, price, and dates.
* **Component Interfaces (internal and external):**

When a patient purchase an insurance plan, a bill with the monthly cost is sent to the patient.

When a doctor finished an appointment, a doctor’s bill get sent to the insurance provider and is automatically paid down. And a bill for the deductible is sent from the insurance provider to the patient.

An unpaid bill will be indicated as pending with a ‘p’ mark.

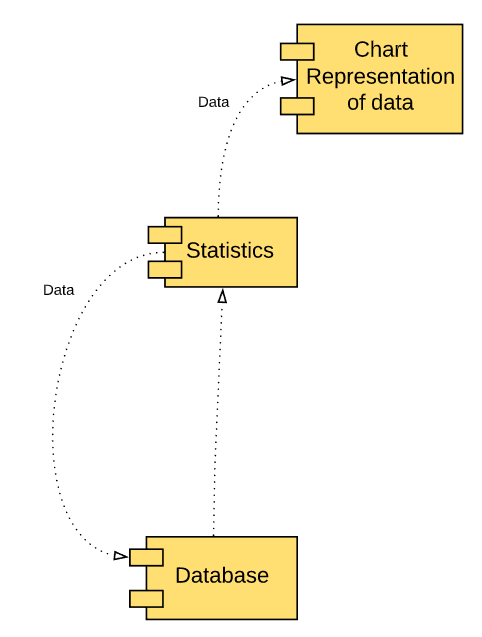
* **Component Error Handling:** It requires further analysis.

10.

* **Component Name:** Statistics
* **Component Description:** Using this component, users can see relevant information and statistics.

# Responsible Development Team Member: Aravind, Eugene

* **Component Diagram:**

****

* **Component User Interface:** After the user signs in , he can access the statistics page by clicking on the statistics tab on the left hand side.
* **Component Objects:** tables and charts to represent statistical data
* **Component Interfaces (internal and external):**

For insurance provider: number of active plan subscribers, average monthly price of plan, monthly revenue chart, monthly insurance payout, month-over-month revenue growth.

For doctor: chart for monthly income and number of appointments

For patient: Amount saved with insurance plan (insurance payout)

All data are either queried from the database or calculated from queried data

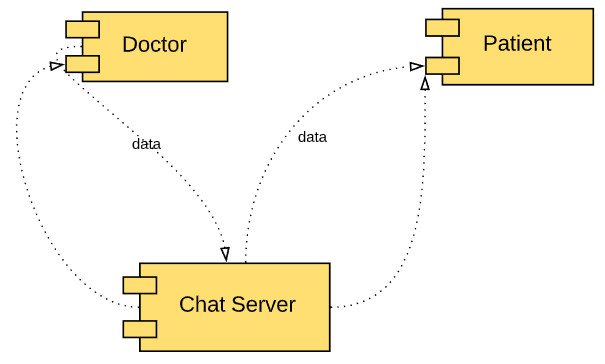
* **Component Error Handling:** It requires further analysis.

11.

* **Component Name:** Chat
* **Component Description:** Using this component, users communicate with each other

# Responsible Development Team Member: Aravind

* **Component Diagram:**

****

* **Component User Interface:** Patient user can search any insurance provider and doctor and initiate a chat with any of them through their profile page. Insurance provider and doctor can only chat with patients they have connections with.
* **Component Objects:** Chat module
* **Component Interfaces (internal and external):**

In the chat window, the left hand side will show a list of active chat rooms with different users. Users can select a chatroom and type in a form then hit enter to send the message. The receiving user will see the message instantly.

* **Component Error Handling:** It requires further analysis.

# Revision History

|  |  |  |
| --- | --- | --- |
| Revision | Date | Change Description |
| Added components – forgot password, tfa, captcha,  and scheduler | 10/19/2019 | Catching up on sprint 1 and adding sprint 2 features. |
| Added Insurance Purchase and Recommendation | 11/17/2019 | Sprint 4 edit |
| Added Billing, Statistics, and Chat | 12/7/2019 | Sprint 5 edit |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |