MEDICATION TAKE BACK CATALOGING APP

A Major Project Report

Submitted in partial fulfilment of the requirements for the award
of the degree of

MASTER OF SCIENCE

In

School of Computing and Engineering

By

Veeresha M Thotigar	**53**
Sai Sampath Kumar Raigiri	**41**
Vineetha Gummadi	**15**
Vidyullatha Lakshmi Kaza	**26**

Submission Date: 12/12/2018

Under the supervision of

Dr. Yug Yung Lee

University of Missouri-Kansas City

5100 Rockhill Rd, Kansas City, MO 64110

TABLE OF CONTENTS

INTRODUCTION

Unused home medications serve as a source for poisonings, abuse, and misuse. Leftover medications include two primary categories: over-the-counter and prescription products, with prescription further classified as non-controlled or controlled substances.

Managing unused, unwanted and expired medications is a safety as well as an environmental concern. Safety and accidental poisoning concerns for smaller children and family pets are on the rise, however, headlines across the nation are focusing on two distinct areas of concern: the contamination of drinking water supplies with pharmaceuticals, and the rise of teen abuse of prescription medications.

Traditionally, we were told to flush unwanted medications down the drain or toilet rather than keeping them in the home. Although effective in preventing medication flushing creates a new and growing problem in the environment. Antibiotics and other medications in a septic system can destroy beneficial bacteria necessary for the system to operate. Wastewater treatment plants are not designed to remove or process many compounds found in medications that end up being discharged into our surface and groundwater.

Community Members, especially parents of teenagers, need a safe and consistent means of disposing of unwanted medications. According to the Monitoring, the Future Survey conducted by the National Institute on Drug Abuse (NIDA), most of the teens reporting the use of medications say that they obtained them from friends or family members, with one-fifth to one-quarter reporting taking them without permission. Parents and caregivers need to understand the importance of safeguarding and proper disposal of their medications.

Here comes our idea “Medication take back events” collect medications from individuals and households. National attention is growing and more appropriate methods of safely disposing of unwanted medications are in the works.

PROJECT GOAL

The motivation of our project is to educate the community and bring awareness to the following issues regarding disposal of unused, unwanted and expired medications from the home and dispose of them safely to prevent poisonings, prevent prescription drug abuse, and protect the environment.

PROJECT OBJECTIVES

- Illustrate the degree to which controlled prescription medications remain unused.
- Identify specific classes/agents that are more likely to remain unused.
- The objective of the system is to record all the features/details of the medications quantity prescribed and quantity remaining at the take-back events and generates the reports for public awareness and research motive.
- Provide recommendations for reducing the accumulation of medications in homes.

FEATURES

Our Medication take-back application can be accessible by all the authorized members. The system will have the features like,

1. Built using an IONIC platform.
2. Can be used on iPhone and Android mobile phones.
3. Tested on Android and IOS platform.
4. Scan the medicine using a camera.
5. Report generation.
6. The generated data will be auto-filled.
7. Generated Data will be saved in the database
8. Gathering the details of the medication.
9. Generation of the report on daily, weekly, monthly ranges and based on the event place.

SIGNIFICANCE

1. To bring awareness among people of safe disposal of unwanted medications.
2. Teen abuse of prescription medications.
3. Safety and accidental poisoning of children and family pets.
4. Keeping our environment clean of toxins from medications.

SCREENS

For Increment - 1 we have developed the following pages:

1. Login Page
2. Registration Page
3. About Us

For Increment - 2 we have developed the following pages:

1. Events home page
2. Events create page
3. Events join page
4. Product log page

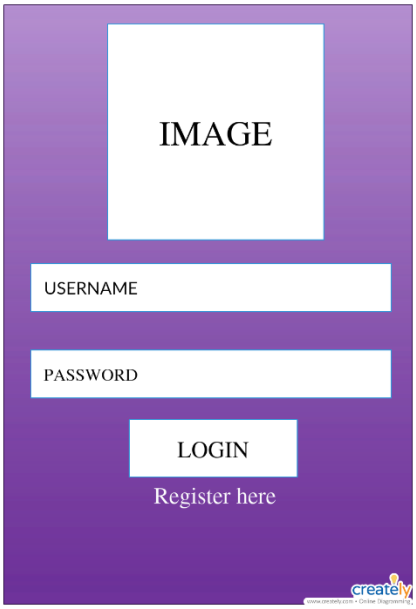
For Increment - 3 we have developed the following pages:

1. Chat Page
2. Barcode Generation
3. Event Page
4. Report page

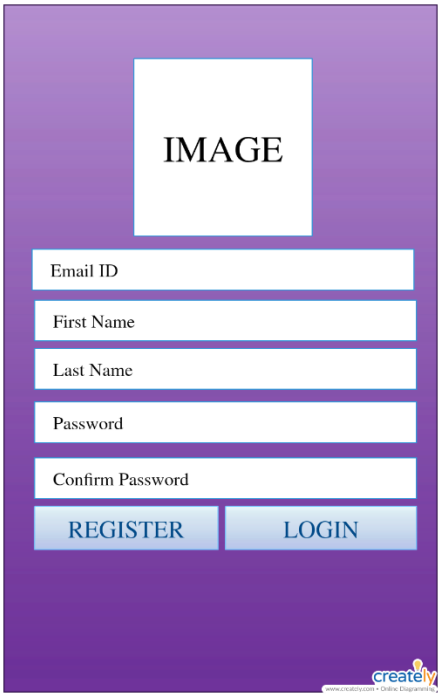
DETAIL DESIGN OF FEATURES (Using Tools)

Wireframes and Mockups:

Login Page



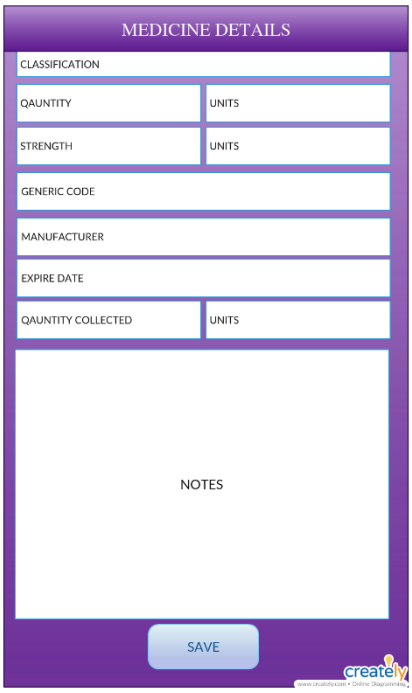
Registration Page



Home Page

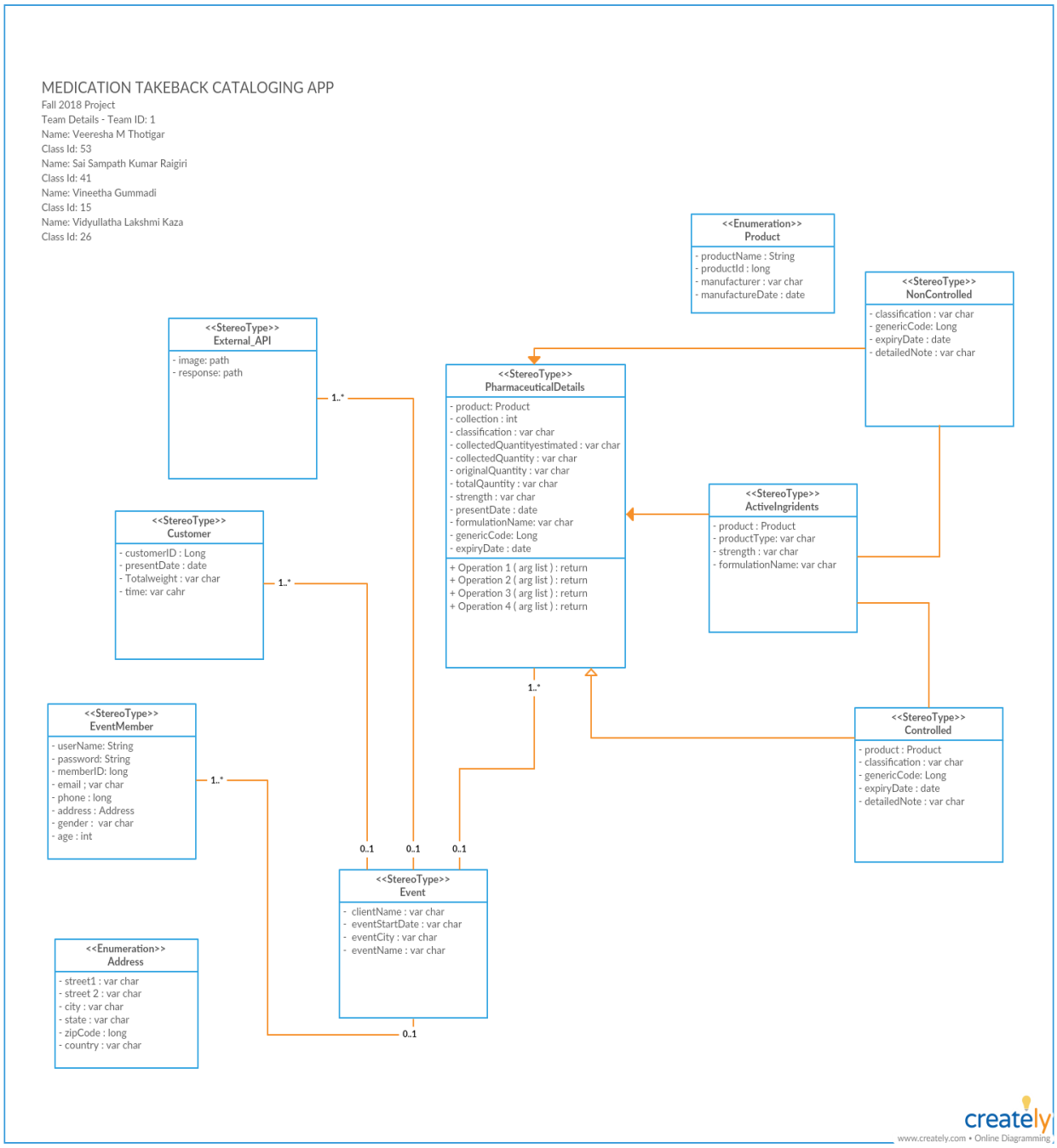

Medication Take Back
LOGOUT
IMAGE
SCAN PILLL


MEDICINE DETAILS
IMAGE
PRODUCT INFORMATION
NAME OF THE PRODUCT
COLLECTION #
CLASSIFICATION
QAUNTITY
UNITS
STRENGTH
UNITS
GENERIC CODE
MANUFACTURER

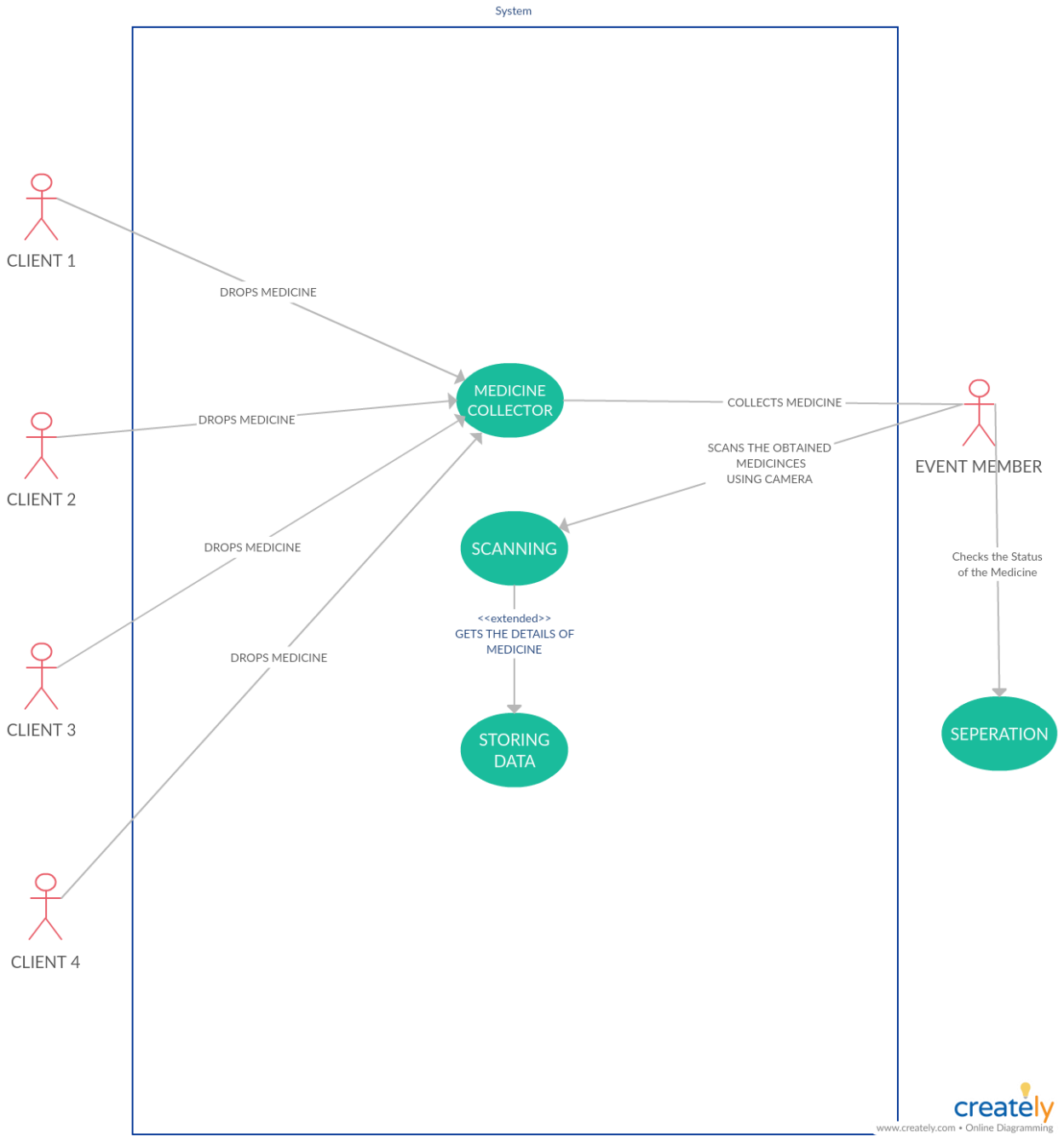


After capturing the picture. All the required details about the medicine will be autofilled from REST services.

Architecture diagram/Sequence diagram/Class diagram



Write User Stories /Use Case/Service description



Implementation (using Ionic 3,Angular, Node Js,MongoDB)

Fire Base Page

In Ionic 3, authentication and registering the application users is done by using the firebase service. Below is the code snippet to initialize the firebase.

```
//firebase
import { AngularFireAuthModule } from "angularfire2/auth";
import { AngularFireDatabaseModule } from "angularfire2/database";
import { AngularFireModule } from "angularfire2";

//barcodescanner
import { BarcodeScanner } from '@ionic-native/barcode-scanner';

// Initialize Firebase
export const firebaseConfig =
{
  // insert config here

};
```

Login Page

Login page form provides the input field for username and password, the entered credentials are validated on function calling SignIn(). Below code snippet: left form page and at the right logic to validate.

```
<div class="login-box">

        <ion-list >
          <ion-item>
            <ion-label floating>Username</ion-label>
            <ion-input  type="text" #username required></ion-input>
          </ion-item>
          <ion-item>
            <ion-label floating>Password</ion-label>
            <ion-input  type="password" #password required></ion-input>
          </ion-item>
        </ion-list>
        <div style="text-align: center">
        <button ion-button medium type="submit" (click)="signIn()" round>Login</button>
      </div>
        <button ion-button block clear (click)="register()">Create New Account</button>


</div>
```

```
signIn(){
  var validate = true

  if(this.uname.value == "" || this.pwd.value == ""){
    alert("Please enter the credentials")
    validate= false
  }
  if (validate == true){
    this.firebase.auth.signInWithEmailAndPassword(this.uname.value, this.pwd.value).the
        console.log("Sign in successful");
        this.navCtrl.push(TabsPage);
    }).catch(error => {
      console.log("error in authentication : ", error);
      console.log(error.message);
    });
  }
}

register(){
this.navCtrl.push(RegisterPage)
}
```

User Registration Page

Registration page form provides the input field for username details, the entered details are validated and saved on function calling register(). Below code snippet: shows left form page and at the right logic to validate and save details.

```
<div *ngFor="let event_obj of result" style="width:100%">
<ion-card  >
    <ion-item>
      <ion-avatar item-start>
        <img src="https://www.aegon.co.uk/content/dam/ukpaw/advisers-login.svg">
      </ion-avatar>
      <h2>{{event_obj.eventName}}</h2>
      <p>{{event_obj.eventStartDate | date:'short'}}</p>
    </ion-item>
    <img (click)="event(event_obj.eventName,event_obj.eventStartDate,event_obj.address_city,event_obj.address
    <ion-card-content>
      <p>
      </p>
    </ion-card-content>
    <ion-row>
      <ion-col >
        <ion-item>
          <button ion-button icon-start clear small >
            <ion-icon name="chatbubbles"></ion-icon>
            <div>chat</div>
          </button>
        </ion-item>
      </ion-col>
      <ion-col  >
          <ion-item style="width:100%; float: right">
            <button  ion-button icon-start clear small end>
              <ion-icon name="more" ></ion-icon>
              <p>details </p>
            </button>
        </ion-item>
      </ion-col>
```

```
register(){
  var validate = true
  if(this.emailId.value == "" || this.firstName.value == "" || this.lastName.value =="|| this.pwd.value == "
  {
    alert("Fields should not be empty" )
    validate = false;
  }
  if(!(this.pwd.value == this.confirmPwd.value))
  {
    alert("Password Should Match" )
    validate = false;
  }
  if (validate == true){
    this.url = 'http://127.0.0.1:3000/users/create';
    this.firebase.auth.createUserWithEmailAndPassword(this.emailId.value, this.pwd.value).then(data =>{
        console.log("Got data from Firebase: ", data);
        this.http.post(this.url,{
          email:this.emailId.value,
          firstName:this.firstName.value,
          lastName:this.lastName.value
        }).subscribe(
          (res:any)=>{
            this.message = res.message;
            alert(this.message);
            this.navCtrl.push(loginPage);
          }
        )
    }).catch(error =>{
      console.log("error in registration : ", error);
    });
  }
```

Event Creation Page

Event creation page form provides the input field for event details, the entered details are saved on function calling createEvent(). Below code snippet: shows left form page and at the right logic to save details.

```
    <ion-item>
      <ion-label floating>Address 2</ion-label>
      <ion-input type="text" [(ngModel)]="address_two"></ion-input>
    </ion-item>

    <ion-item>
      <ion-label floating>City</ion-label>
      <ion-input type="text" [(ngModel)]="address_city"></ion-input>
    </ion-item>

     <ion-item>
      <ion-label floating>State</ion-label>
      <ion-input type="text" [(ngModel)]="address_state"></ion-input>
    </ion-item>

    <ion-item>
      <ion-label floating>ZIP</ion-label>
      <ion-input type="number" [(ngModel)]="address_zipcode"></ion-input>
    </ion-item>
  </ion-list>
  <br/>
  <br/>
  <div padding style="text-align: center">
    <button ion-button (click)="createEvent()" round medium>Create</button>
  </div>
</ion-content>
```

```
  createEvent(){
    this.url = 'http://127.0.0.1:3000/event/create';
    this.http.post(this.url,{
      eventName:this.eventName,
      eventStartDate:this.eventStartDate,
      eventEndDate:this.eventEndDate,
      address_one:this.address_one,
      address_two:this.address_two,
      address_city:this.address_city,
      address_state:this.address_state,
      address_zipcode:this.address_zipcode,
      users_list:this.userlist,
      created_by:this.created_by
    })
      .subscribe(
        (res:any)=>{
          this.message = res.message;
          alert(this.message);
          this.navCtrl.pop();
          console.log(this.message);
        }
      )
  }
```

Join Event Page

Other events that are created by other users are listed here, if interested the current logged in user can join in the events listed. Below code snippet: shows left list page and at the right logic to join event.

```
<ion-content class="card-background-page">
 <div *ngFor="let event_obj of result">

   <ion-card>
      <img style="height:300%" src="https://www.ctiicc.co.za/sites/default/files/static_images/feature-events1
      <div class="card-title" style="color: #6543ff; font-family: Cambria">
         {{event_obj.eventName}}
      </div>
       <ion-row>
          <ion-col>
            <ion-item>
                <p style="font-size: 20px">When:</p>
                <p>{{event_obj.eventStartDate | date:'short'}}</p>
            </ion-item>
          </ion-col>
          <ion-col>
              <ion-item>
                  <p style="font-size:20px">Where:</p>
                  <p>{{event_obj.address_one}}, {{event_obj.address_two}}</p>
                  <p>{{event_obj.address_city}}, {{event_obj.address_state}}</p>
                  <p>{{event_obj.address_zipcode}}</p>
              </ion-item>
            </ion-col>
       </ion-row>
       <ion-row>
          <ion-col>
            <ion-item>
                <button ion-button clear icon-only (click)="joinTheEvent(event_obj.eventName)">
                    <ion-icon name="addEvent" md="md-add"></ion-icon> Join
                </button>
```

```
loadevents(name:string){
  this.url = 'http://127.0.0.1:3000/events/search/users?user=false&searchtext='+name;
  this.http.get(this.url)
    .subscribe(
      (res:any)=>{
        this.result = res.data;
        console.log(this.result);
      }
    )
}
joinTheEvent(name:string){
  this.url = 'http://127.0.0.1:3000/event/update/users';
  this.http.put(this.url,{
    event_name:name,
    user:this.created_by
  })
    .subscribe(
      (res:any)=>{
        this.message = res.message;
        this.loadevents(this.created_by);
        alert(this.message);
      }
    )
}
}
```

Product search and details save page

On search of the product, its details are retrieved and auto-filled in the form fields, later user can edit and save the product details on function calling drugData(). Below code snippet: shows left form page and at the right logic to save details.

```
<form (ngSubmit)="drugData()" text-wrap >
 <ion-grid><ion-row>
   <ion-col col-12><ion-item>
   <ion-label floating style="font-size: 20px">Name of the Drug or Product</ion-label>
   <ion-input type="text" [(ngModel)]="drugName" name="drugname"></ion-input>
 </ion-item></ion-col>
 </ion-row></ion-grid>
 <ion-grid><ion-row>
   <ion-col col-12><ion-item>
   <ion-label floating style="font-size: 20px">Description</ion-label>
   <ion-textarea [(ngModel)]="drugDescription" name="description"></ion-textarea>
   </ion-item></ion-col>
 </ion-row></ion-grid>
   <ion-grid><ion-row>…
   </ion-row></ion-grid>

   <ion-grid><ion-row>…
   </ion-row></ion-grid>

   <ion-grid><ion-row>…
  </ion-row></ion-grid>

   <ion-grid><ion-row>…
  </ion-row></ion-grid>

  <div style="text-align: center">
 <button ion-button type="submit" round>Save</button>
</div>
</form>
```

Server Side: REST API's

App: MongoDB connection, Schema initialization, controller imports

In Node JS, app.js is the main server file where express is initialized and incorporates the CORS. A logic for local database connection, schema initialization and controller imports are as shown below.

app.js

```javascript
var createError = require('http-errors');
var express = require('express');
var path = require('path');
var cookieParser = require('cookie-parser');
var logger = require('morgan');
const db = require('mongoose'),db_string = 'mongodb://localhost:27017/medication_take_back';
var app = express();
var cors = require('cors');

//initializing schema
require('./model/drug');
require('./model/events');
require('./model/users');
require('./model/users_seq');
//connection for DB
var db_promise = db.connect(db_string,{ useNewUrlParser: true });
db_promise.then((data) => {
// view engine setup
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'ejs');

app.use(logger('dev'));
app.use(express.json());
app.use(express.urlencoded({ extended: false }));
app.use(cookieParser());
app.use(express.static(path.join(__dirname, 'dist')));
app.use(cors());

// Rest APIs
require('./controllers/drug')(app, db);
require('./controllers/events')(app, db);
require('./controllers/users')(app, db);

// catch 404 and forward to error handler
app.use(function(req, res, next) {
  next(createError(404));
});
```

User Controller:

GET: /users/search

POST: /users/create

```
app.js  drug.js  events.js  users.js
2   module.exports = function (app, db) {
3       let users_model = db.model('users');
4       let users_seq_model = db.model('users_seq');
5       //api to search user details
6       app.get('/users/search',(req,res)=>{
7           let search_text = req.query.searchtext;
8           let search_by = req.query.searchby;
9           let query = {};
10          query[search_by] = { $regex: search_text, $options: 'i' };
11          users_model.find(query).exec((err, users) => {
12              if (!err) {
13                  res.send({
17              } else {
18                  res.status(400).send({
23              }
24          });
25      });
26      //api to create user details
27      app.post('/users/create',(req,res) => {
28          let users_info = req.body;
29          users_seq_model.findOneAndUpdate({ seq_id : "userid" },{$inc:{sequence_value:1}},{new:true,useFindAndModify:false},(err,item)
30              if(err) throw err;
31              if(!item){
32                  let seq = new users_seq_model({seq_id:"userid",sequence_value:1});
33                  seq.save((err,obj)=>{
59              }else{
60                  console.log(item)
61                  let users = new users_model({
69                  users.save((err, users_res) => {
70                      if (!err) {
71                          res.send({
75                      } else {
76                          res.status(400).send({
81                      }
82                  })
83              }
```

Event Controller:

GET: /events/search/users

POST: /events/create

PUT: /event/update/users

```
app.js  drug.js  events.js
1
2   module.exports = function (app, db) {
3       let events_model = db.model('events');
4       //api to search events details
5       app.get('/events/search/users',(req,res)=>{
6           let user = req.query.user;
7           let search_text = req.query.searchtext;
8           let query = {};
9           if(user === 'true'){
10              query = {users_list:{$in:[search_text]}};
11          }else{
12              query = {users_list:{$nin:[search_text]}};
13          }
14          events_model.find(query).exec((err, events) => {
28      });
29      //api to create event details
30      app.post('/event/create',(req,res) => {
31          let events_info = req.body;
32          let events = new events_model({
46          events.save((err, events_res) => {
60      });
61      //api to update event details
62      app.put('/event/update/users',(req,res)=>{
63          let events_info = req.body;
64          events_model.updateOne({'eventName':events_info.event_name},{$push:{'users_list':events_info.user}},(err, events_res) => {
78      });
79  };
```

Product Controller:

GET: /drug/search

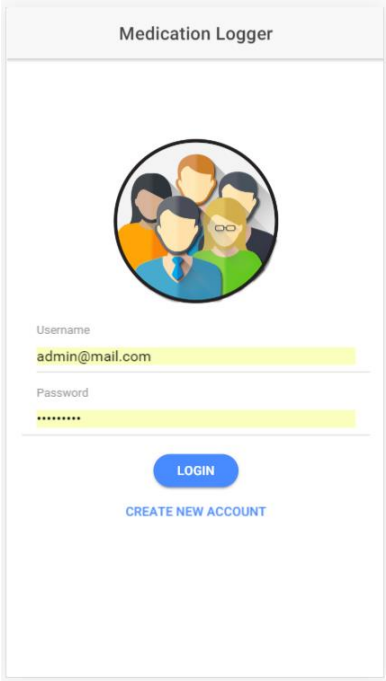
POST: /product/create

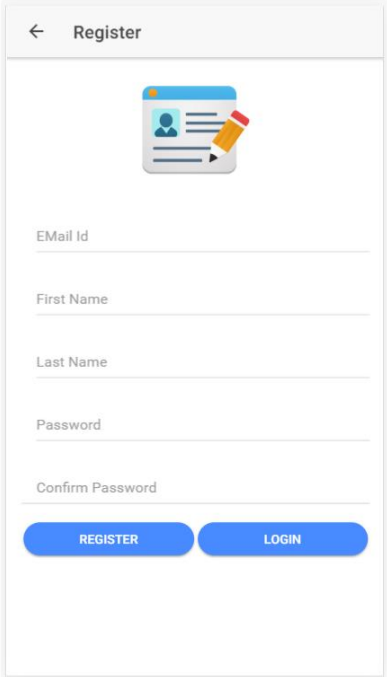
app.js drug.js

```
1
2  module.exports = function (app, db) {
3      let drug_model = db.model('drug');
4      let product_model = db.model('products');
5      //api to search drug details
6      app.get('/drug/search',(req,res)=>{
7          let search_text = req.query.searchtext;
8          let query1 = {"name":{ $regex: search_text, $options: 'i' }};
9          let query2 = {"description":{ $regex: search_text, $options: 'i' }};
10         let mainQuery = {$or:[query1,query2]};
11         drug_model.find(mainQuery).exec((err, drug) => {
12             if (!err) {
13                 res.send({
17             } else {
18                 res.status(400).send({
23             }
24         });
25     });
26
27     //api to create Product details
28     app.post('/product/create',(req,res) => {
29         let product_info = req.body;
30         let product = new product_model({
47
48         product.save((err, prod_res) => {
49             if (!err) {
50                 res.send({
54             } else {
55                 res.status(400).send({
60             }
61         })
62     });
63 };
```

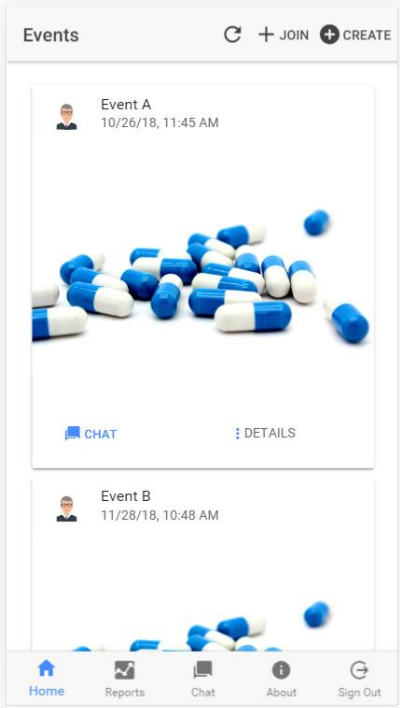
Outputs:

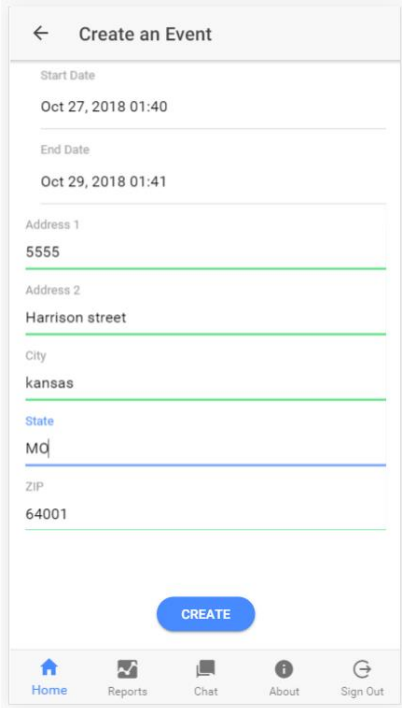
Login and Register

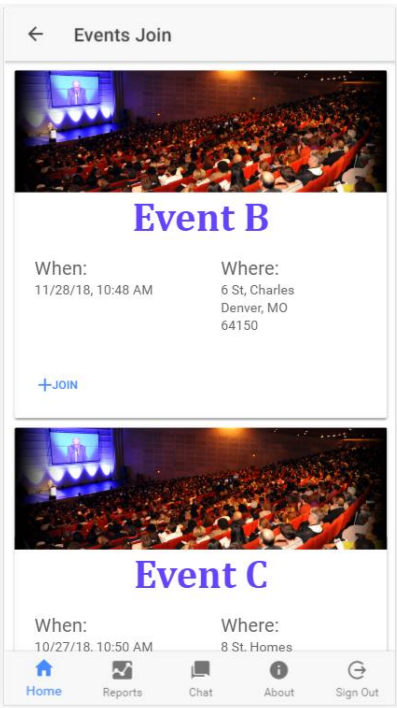




EVENT

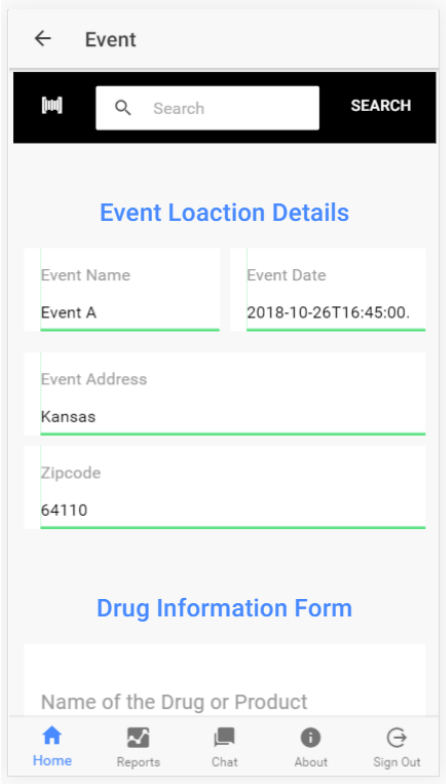


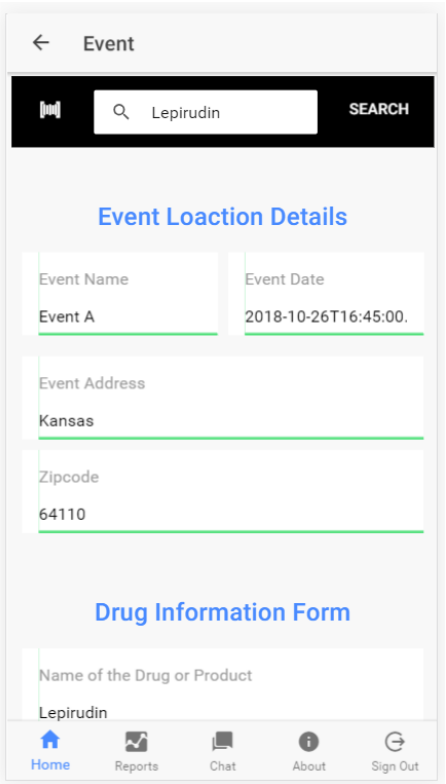




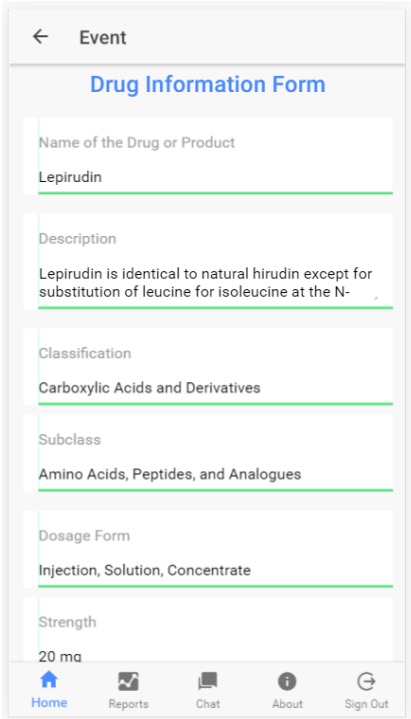
Product Log Details

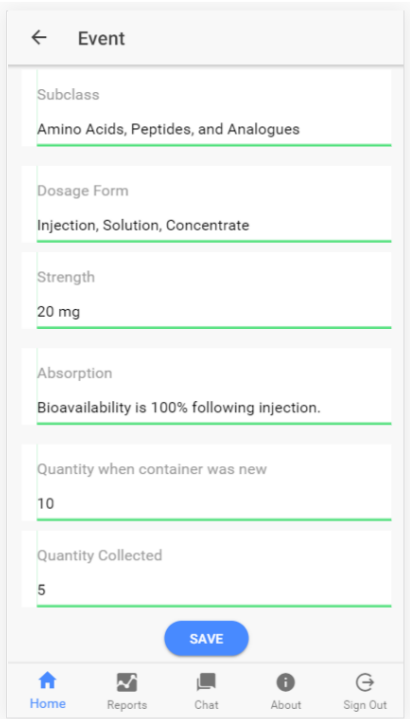
Product Search



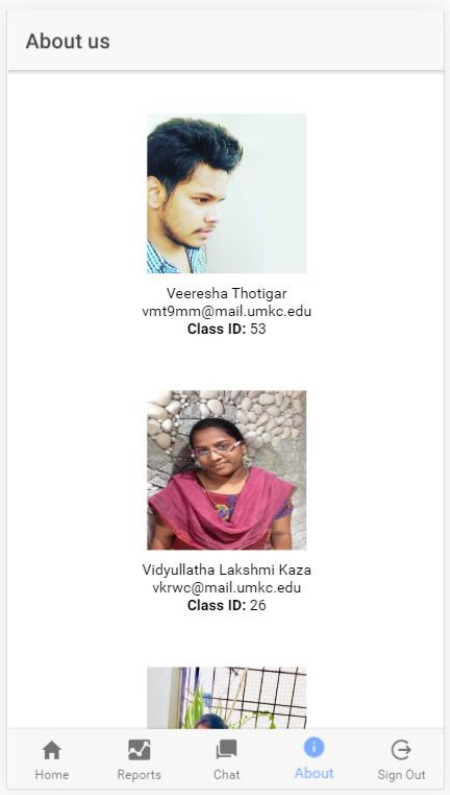


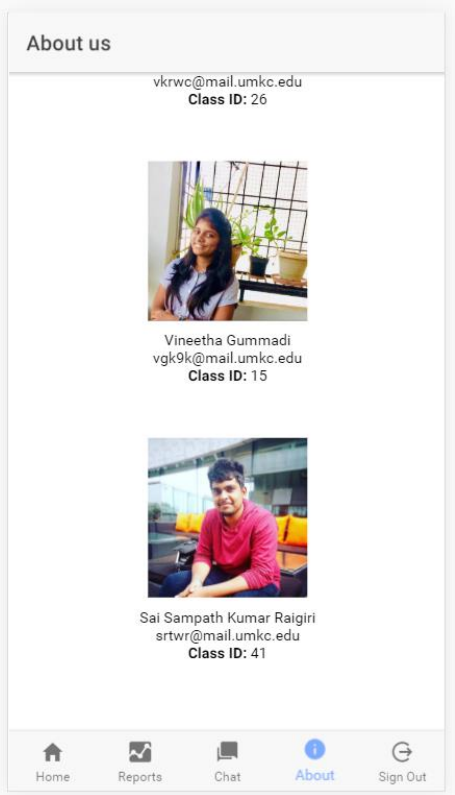
Product details Log



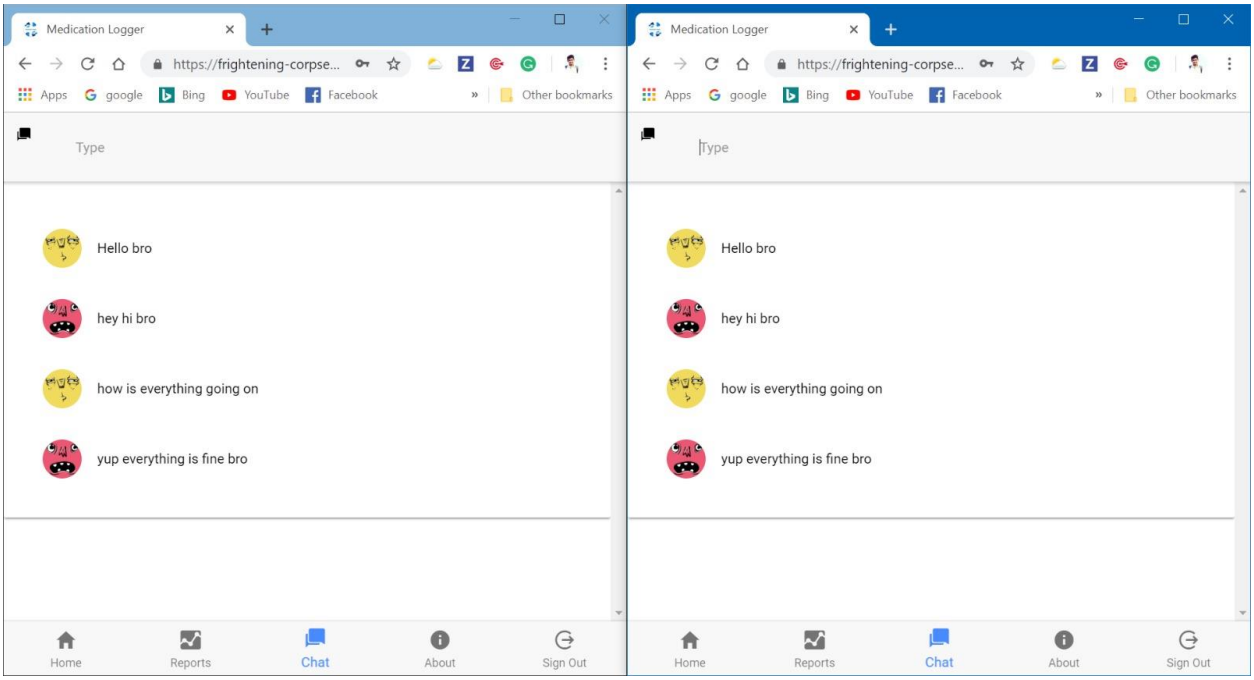


About Page

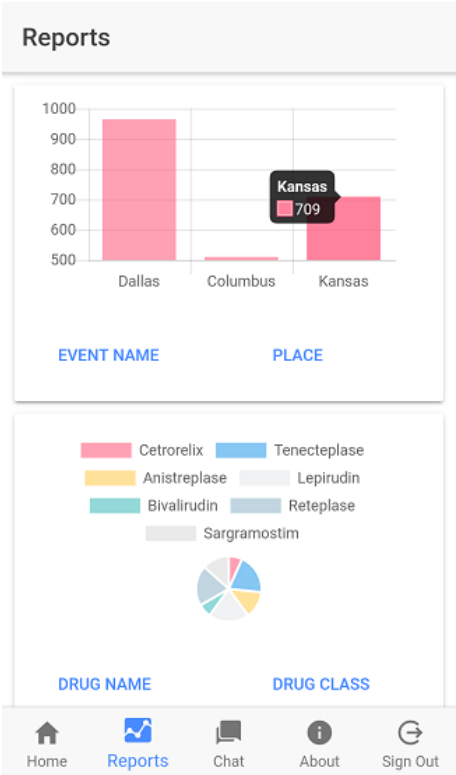


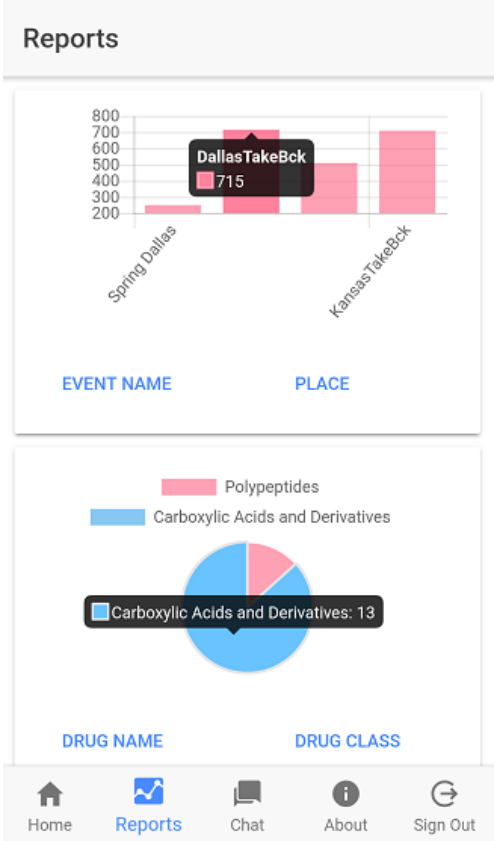


Chat Page



Report generation



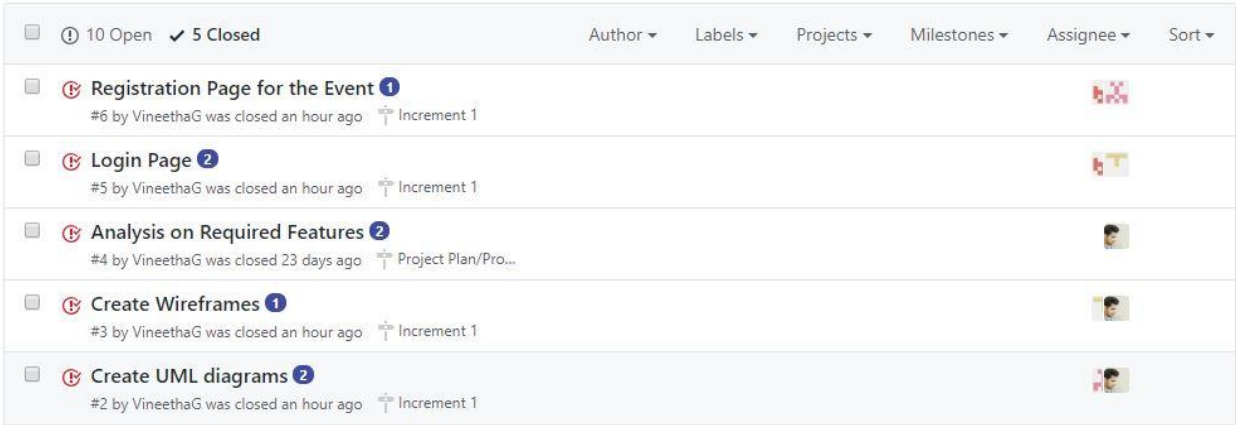


Project Management: describe it in terms of the first increment (with ZenHub):

Implementation status report

Work completed

Create Issues/tasks for Increment 1:



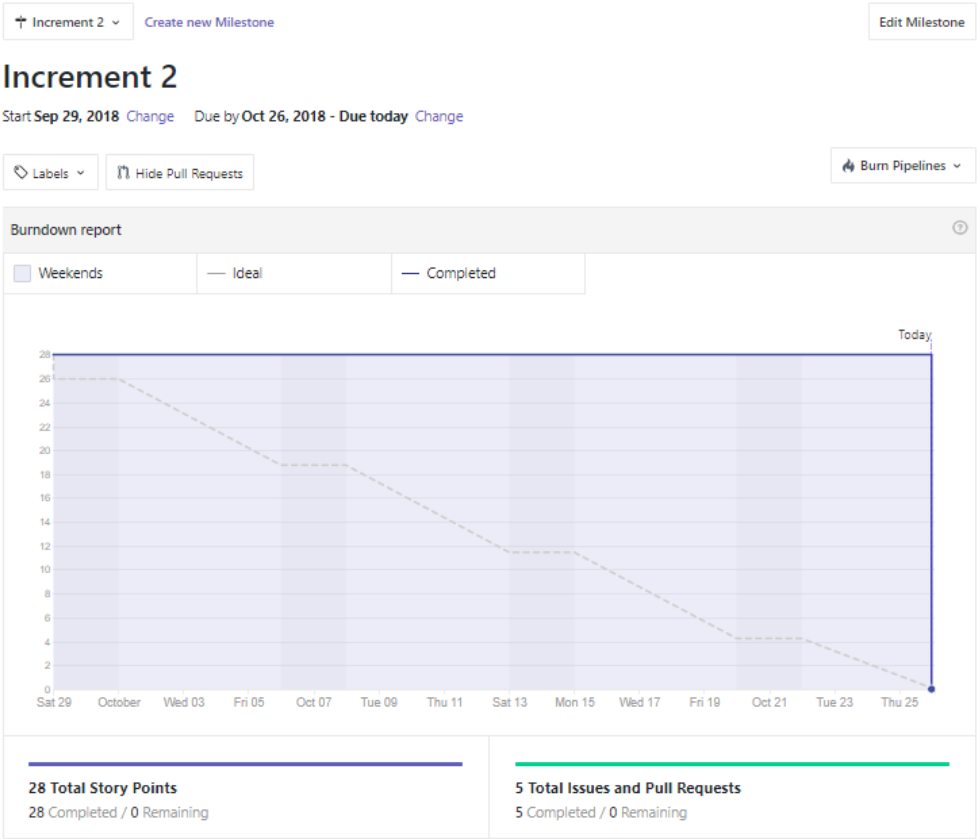
Burndown Chart for increment-1

Increment 2

Create Issues/tasks for Increment 2:

Completed Issues and Pull Requests	Story points
Basic server and Database setup CS5551_WebKrakers_ProjectTeam1 #12 Closed Increment 2	5
RESTFUL API's for login and register CS5551_WebKrakers_ProjectTeam1 #13 Closed Increment 2	5
RESTFUL API's for data collection CS5551_WebKrakers_ProjectTeam1 #14 Closed Increment 2	6
Client app development for data collection CS5551_WebKrakers_ProjectTeam1 #15 Closed Increment 2	6
Client app development mashup container CS5551_WebKrakers_ProjectTeam1 #16 Closed Increment 2	6

Burndown chart for Increment 2:

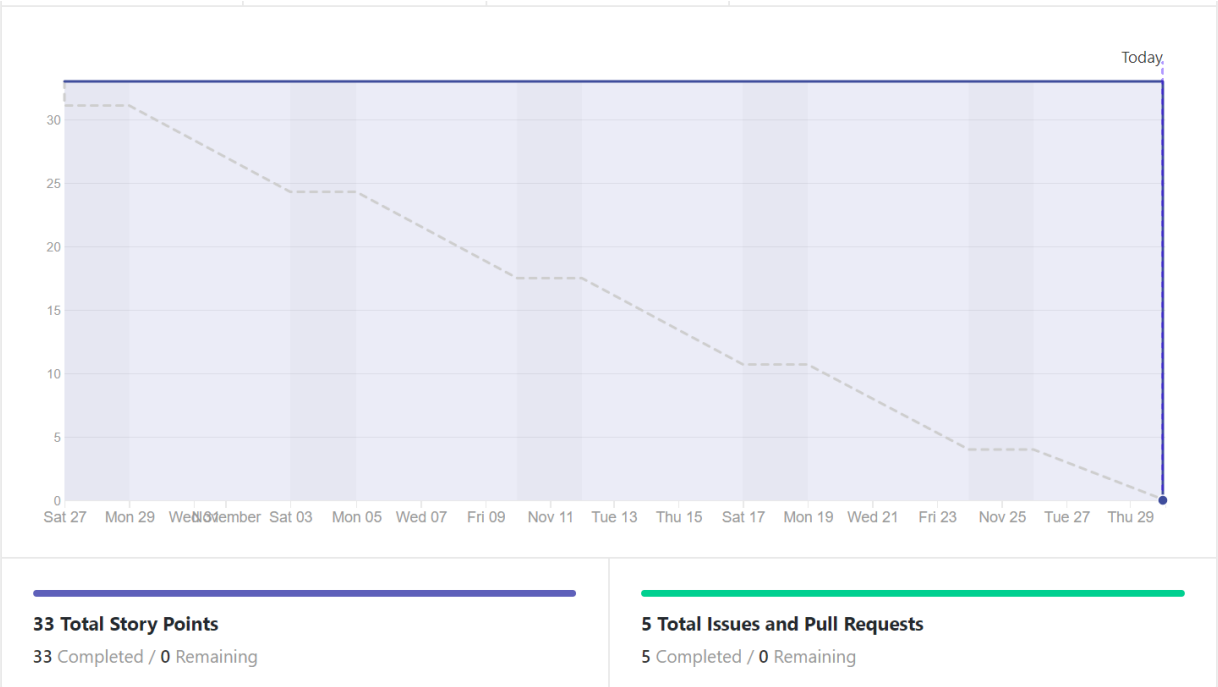


Increment 3

Create Issues/tasks for Increment 3:

Completed Issues and Pull Requests	Story points
Testing and fixing the defects CS5551_WebKrakers_ProjectTeam1 **#7** Closed Increment 3	5
Restful API's for reports part-1 CS5551_WebKrakers_ProjectTeam1 **#17** Closed Increment 3	7
Restful API's for reports part-2 CS5551_WebKrakers_ProjectTeam1 **#18** Closed Increment 3	7
Client side dashboard development - 1 CS5551_WebKrakers_ProjectTeam1 **#19** Closed Increment 3	7
Client side dashboard development - 2 CS5551_WebKrakers_ProjectTeam1 **#20** Closed Increment 3	7

Burndown chart for Increment 3:



URL:

Project URL:

https://github.com/saisampathkumar/CS5551_WebKrakers_ProjectTeam1

Project Video URL:

https://www.youtube.com/watch?v=rcifpz4VUIE&feature=youtu.be

Android Deployment Link:

https://github.com/saisampathkumar/CS5551_WebKrakers_ProjectTeam1/blob/master/APK/app-debug.apk

Cloud Deployment Link:

https://medication-take-back.herokuapp.com/

CONCLUSION

- Using this application, the process at take back event is made efficient and procedural.
- Visualizing the reports gives insight to policy makers to decide the laws on over prescribed drug.

BIBLIOGRAPHY

1. https://ionicframework.com/docs/
2. https://nodejs.org/en/
3. https://www.mongodb.com/
4. https://www.drugbank.ca/
5. https://takebackday.dea.gov/?src=deatakeback.com
6. https://www.texaspain.org/assets/3-%20Jaramillo%20TPS%20Presentation.pdf
7. https://www.tandfonline.com/doi/full/10.1080/14659891.2017.1337821
8. https://texansstandingtall.org/pdfs/Rx%20takeback%20final.pdf
9. http://www.idmypill.com/api/
10. https://lhncbc.nlm.nih.gov/rximage-api