

PPA Membership Maintenance System

Sprint Report 2

Capstone Computing Project 2

Group SD07

Semester 2, 2018

Curtin University – Department of Computing

Assignment Cover Sheet / Declaration of Originality

Complete this form if/as directed by your unit coordinator, lecturer or the assignment specification.

Last name:	Ronaka	Student ID:	19211817
Other name(s):	Janith		
Unit name:	Capstone Computing Project 2	Unit ID:	ISAD3001
Lecturer / unit coordinator:	Dr. Hannes Herrmann	Tutor:	Ms. Geethanjalie Wimalarathne
Date of submission:	31/08/2018	Which assignment?	Sprint Report 2

I declare that:

- The above information is complete and accurate.
- The work I am submitting is *entirely my own*, except where clearly indicated otherwise and correctly referenced.
- I have taken (and will continue to take) all reasonable steps to ensure my work is *not accessible* to any other students who may gain unfair advantage from it.
- I have *not previously submitted* this work for any other unit, whether at Curtin University or elsewhere, or for prior attempts at this unit, except where clearly indicated otherwise.

I understand that:

- Plagiarism and collusion are dishonest, and unfair to all other students.
- Detection of plagiarism and collusion may be done manually or by using tools (such as Turnitin).
- If I plagiarise or collude, I risk failing the unit with a grade of ANN ("Result Annulled due to Academic Misconduct"), which will remain permanently on my academic record. I also risk termination from my course and other penalties.
- Even with correct referencing, my submission will only be marked according to what I have done myself, specifically for this assessment. I cannot re-use the work of others, or my own previously submitted work, in order to fulfil the assessment requirements.
- It is my responsibility to ensure that my submission is complete, correct and not corrupted.

Signature: _____



Date of signature: 27/08/2018

(By submitting this form, you indicate that you agree with all the above text.)

Table of Contents

1. Introduction	2
1.1 Group Introduction	2
1.2 Project Introduction.....	2
2. Progress Update.....	3
2.1 Allocated Tasks for the Sprint 2	3
2.2 Planned Tasks for the Sprint 3	3
3. Task Break Down.....	4
3.1 Task 13.1	4
3.2 Task 13.2	5
3.3 Task 26.1	10
3.3 Task 26.2	11
3.4 Task 26.3	14
3.5 Task 26.4	15
3.5 Task 26.5, 26.6, 26.7	18
3.7 Task 8	24
3.8 Task 6.2	25
3.9 Task 7.2	25
4 Development Methodology.....	26
4.1 Minutes	26
4.2 Burndown Chart.....	27
4.3 Sprint Retrospective (Task 7.2)	28
4.4 Time Management.....	29

1. Introduction

1.1 Group Introduction

All the members in our group have successfully completed the CCP1 in 2017. We enrolled for the CCP2 module in 2018 2nd semester, therefore this project is started in semester 2 of this year (2018). Because of that we have to done the main documentations and tasks of the project such as SRS, task allocation, initial requirement gathering etc. in semester 2. Each of the group members has to do a workload of 2 semesters within this semester, in order to complete the project successfully.

1.2 Project Introduction

PPA Membership Maintenance System is going to be used for membership management and some other important administration tasks such as event planning, donation collecting, accounts handling etc. of past pupil association of Sirimavo Bandaranaike Vidyalaya. Currently all these operations are manually performed by the committee. The main intention of the system is to automate most of those tasks and perform the semi-automated tasks easily and conveniently.

We use MEAN stack to develop this application, JIRA as the project management tool and bitbucket as the online repository. The application has main 4 parts as Membership Services, Accountings, Event Planning and Reporting. These four sections are interconnected with each other as per their functionalities.

2. Progress Update

Sprint 2: 18th August – 31st August

2.1 Allocated Tasks for the Sprint 2

Task ID	Task	Task Status
Task 13.1	Admin Dashboard - Main GUI Designing	Completed
Task 13.2	Admin Dashboard - Main GUI Implementation	Completed
Task 26.1	News Screen GUI Design	Completed
Task 26.2	News Screen GUI Implementation	Completed
Task 26.3	News Screen Database Design	Completed
Task 26.4	News Screen Validation Methods	Completed
Task 26.5	News Screen Insert Methods	Completed
Task 26.6	News Screen Update Methods	Completed
Task 26.7	News Screen Delete Methods	Completed
Task 7.2	Sprint 2 Retrospective	Completed
Task 6.2	Planning the user stories for the sprint 3	Completed
Task 8	Design Meetings	In Progress

2.2 Planned Tasks for the Sprint 3

Task ID	Task	Time Estimation
Task 13.3	Admin Dashboard - Summary View Methods	3h
Task 14.1	New Membership Page Designing	2h
Task 14.2	New Membership Step 1 Implementation	2h
Task 14.3	New Membership Step 2 Implementation	2h
Task 14.4	New Membership Step 3 Implementation	2h
Task 14.5	New Membership Step 4 Implementation	2h
Task 14.6	New Membership Page Validation Methods	2h
Task 14.7	New Membership Page Insert Methods	2h
Task 15.1	All membership requests screen - GUI	3h
Task 7.3	Sprint 3 Retrospective	0.5h
Task 6.3	Planning the user stories for the sprint 4	1h
Task 8	Design Meetings	1h

3. Task Break Down

3.1 Task 13.1

(Commits: [3b440fa](#))

Admin Dashboard - Main GUI Designing

Estimate Time: 2 Hours

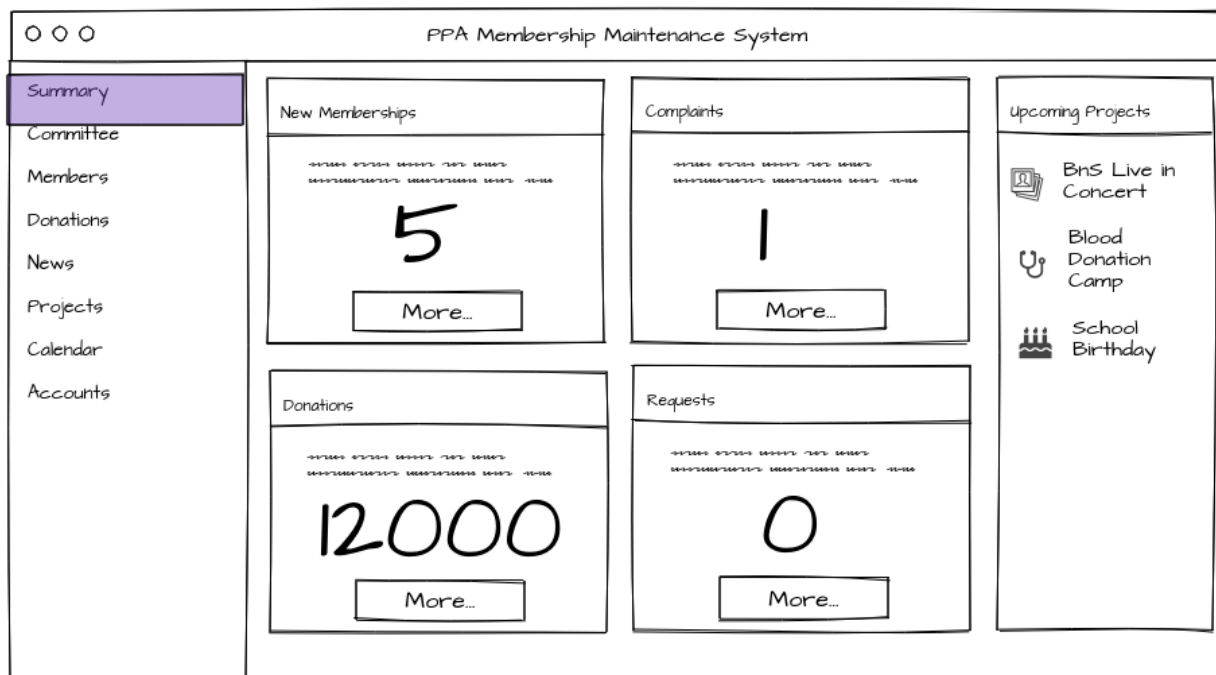
Actual Time: 1 Hour

Actual Time (this sprint): 1 Hour

Description

Since our project is started in this semester (semester 2, 2018) we have to design thinking also during the period of development phase. Under this task I did some thinking about how the administrator dashboard should look like. After some various UI designs finally I came up with the following wireframe.

Since the dashboard should facilitate the admin to easily navigate to the admin features I thought a sidebar panel is the most convenient way to facilitate an easy navigation through the features. I drafted the Upcoming Projects list in the right side of the summary tab so the admins can easily see the most recent projects and navigate to them by just clicking on their names. The reason the



keep the numerical values of the summary section so big is those are the informative factors of the entire summary section. In another word, those are the data that add value to the summary tab. So they should be easily visible to the admin at a glance. If the admin user needs to see more information about those numerical values they can simply click on the button under those values and navigate to the relevant feature screen.

3.2 Task 13.2

(Commits: [3b440fa](#), [13371bf](#))

Admin Dashboard - Main GUI Implementation

Estimate Time: 3 Hours

Actual Time: 4 Hours




Actual Time (this sprint): 4 Hours

Commits & Build Reports


- [3b440fa](#)
- [13371bf](#)

SLIIT Computing Projects / 2018_SD07




Commits

All branches ▾		Find commits			
Author	Commit	Message	Date	Builds	
 Janith Ronaka	13371bf	PPA-238: Main GUI Draft Designing	20 minutes ago		
 Janith Ronaka	3b440fa	PPA-238: Main GUI Draft Designing	4 days ago		

1. [3b440fa – Build Report](#)

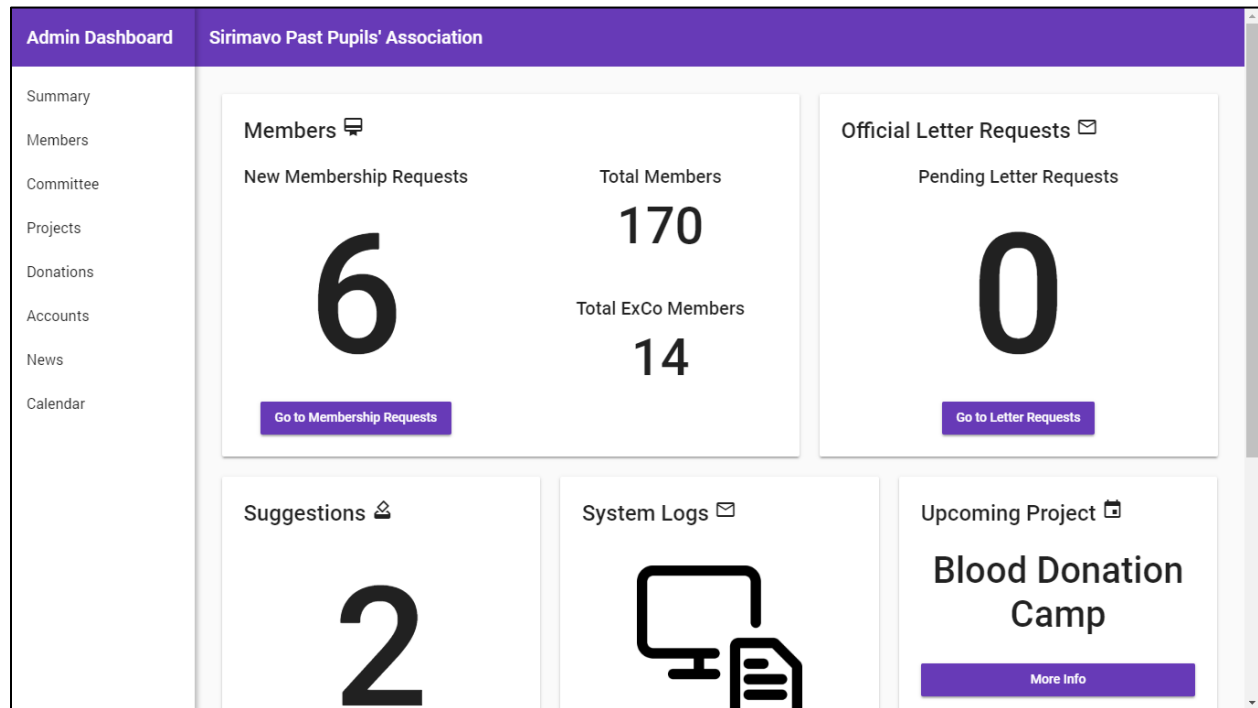
#25  PPA-238: Main GUI Draft Designing  **Successful** 3 days ago 1 min 12 sec
Janith Ronaka  3b440fa  master

2. [13371bf – Build Report](#)

Pipeline	Status	Started	Duration
#27  PPA-238: Main GUI Draft Designing  Successful 4 minutes ago 1 min 13 sec Janith Ronaka  13371bf  master			

Description

When implementing the drafted wireframe, I did some changes to the layout and added some extra fields to the dashboard such as total member count, ExCo member count, System Logs etc. I used different sized card layout for the dashboard.



Technical Information

I used the angular material design components to style the UI. It's far better than the bootstrap and provides a flexible approach. I was able to increase the responsiveness of the interface considering the rendering screen. Following are the some of main components and modules relevant to this implementation.

- admin-sidebar Component
- admin-dashboard Component
- summary-view Component
- app-routing Module

admin-sidebar Component

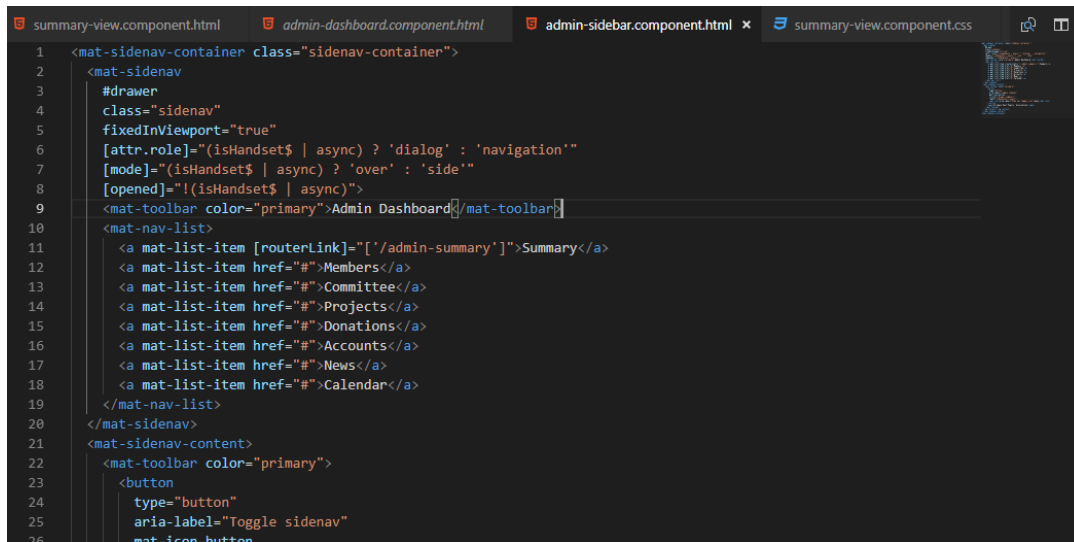
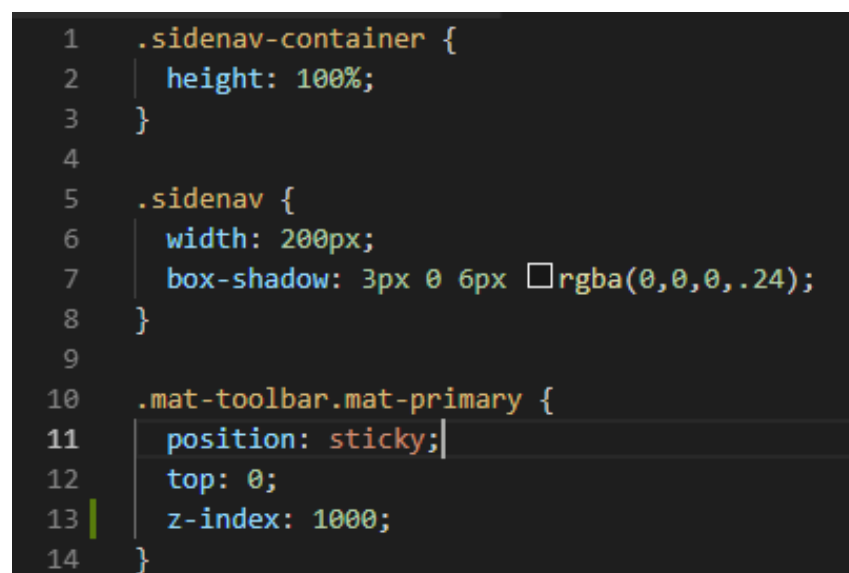


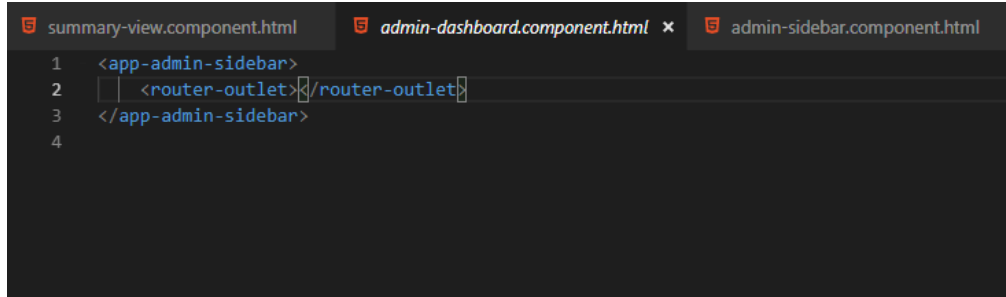
Figure 1: admin-sidebar.component.html

When creating the sidebar I have used the components `mat-sidenav`, `mat-sidenav-container`, `mat-sidenav-content` and `mat-toolbar`. When rendering the `sidenav` I check whether the application is loaded on a handheld device or regular browser screen by subscribing to the `isHandset$` observable using the `async` pipe. If the application is on a handheld device the mode of the `sidenav` is changed to 'over' mode and if it is not the 'side' mode used as the default. Also if the application is not on a handheld device the `sidenav` collapse and shows a toggle button to expand it.

I added the following styles to the `css` file.



admin-dashboard Component

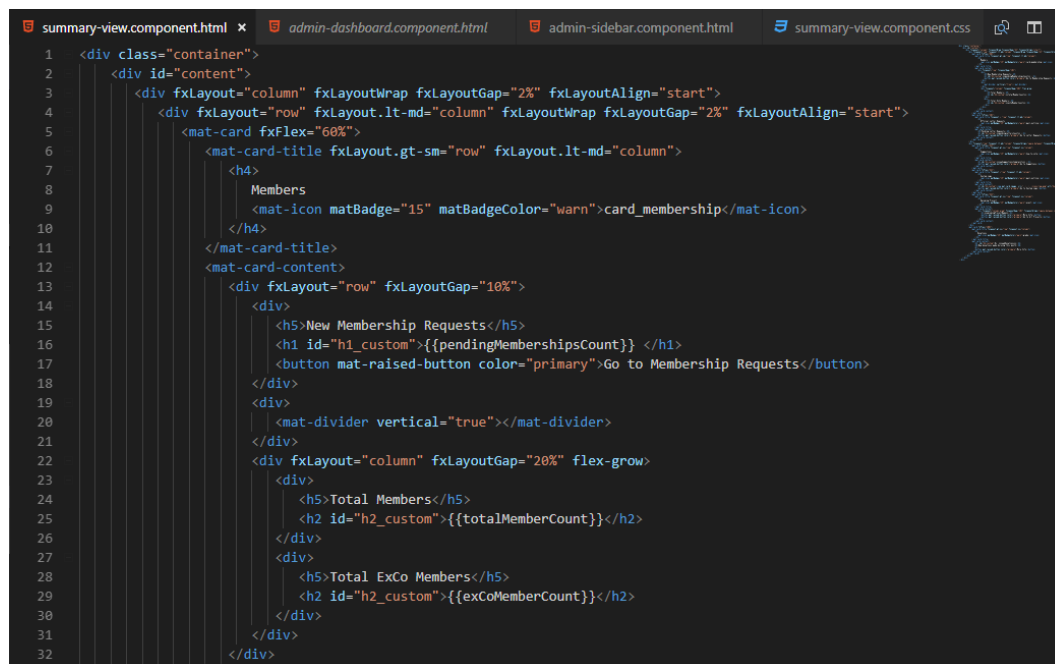


```
1 <app-admin-sidebar>
2   <router-outlet>
3 </app-admin-sidebar>
4
```

Figure 2: admin-dashboard.component.html

This component works as a container to the admin side bar and the relevant tabs of the sidebar. By adding the `<router-outlet>` selector in between of `<app-admin-sidebar>` selector, when use clicked on any of the sidebar item, application loads the relevant component in the content section of the side bar component. Therefore sidebar component works as an frame for the loaded component view.

summary-view Component



```
1 <div class="container">
2   <div id="content">
3     <div fxLayout="column" fxLayoutWrap fxLayoutGap="2%" fxLayoutAlign="start">
4       <div fxLayout="row" fxLayout.lt-md="column" fxLayoutWrap fxLayoutGap="2%" fxLayoutAlign="start">
5         <mat-card fxflex="60%">
6           <mat-card-title fxLayout.gt-sm="row" fxLayout.lt-md="column">
7             <h4>
8               Members
9               <mat-icon matBadge="15" matBadgeColor="warn">card_membership</mat-icon>
10            </h4>
11          </mat-card-title>
12          <mat-card-content>
13            <div fxLayout="row" fxLayoutGap="10%">
14              <div>
15                <h5>New Membership Requests</h5>
16                <h1 id="h1_custom">{{pendingMembershipsCount}} </h1>
17                <button mat-raised-button color="primary">Go to Membership Requests</button>
18              </div>
19              <div>
20                <mat-divider vertical="true"></mat-divider>
21              </div>
22              <div fxLayout="column" fxLayoutGap="20%" flex-grow>
23                <div>
24                  <h5>Total Members</h5>
25                  <h2 id="h2_custom">{{totalMemberCount}}</h2>
26                </div>
27                <div>
28                  <h5>Total ExCo Members</h5>
29                  <h2 id="h2_custom">{{exCoMemberCount}}</h2>
30                </div>
31              </div>
32            </div>
33          </mat-card-content>
34        </mat-card>
35      </div>
36    </div>
37  </div>
```

Figure 3: summary-view.component.html

I used the angular flexbox to layout the material cards in the dashboard. I used the technique two-way data binding to display the numerical data of the dashboard cards. But here I used the two-way data binding as one-way data biding since I don't need to edit the values from the view.

```

1 import { Component, OnInit } from '@angular/core';
2
3 @Component({
4   selector: 'app-summary-view',
5   templateUrl: './summary-view.component.html',
6   styleUrls: ['./summary-view.component.css']
7 })
8 export class SummaryViewComponent implements OnInit {
9
10   pendingMembershipsCount: number = 0;
11   pendingLettersCount: number = 0;
12   newDonations: number = 0;
13   newSuggestionsComplaints: number = 0;
14   upcomingProjectName: string = '';
15   totalMemberCount: number = 0;
16   exCoMemberCount: number = 0;
17
18   constructor() { }
19
20   ngOnInit() {
21     this.getPendingMembershipsCount();
22     this.getPendingLettersCount();
23     this.getNewDonationsAmount();
24     this.getNewSuggestionsComplaintsCount();
25     this.getUpcomingProjectName();
26     this.getTotalMemberCount();
27     this.getExCoMemberCount();
28   }
29

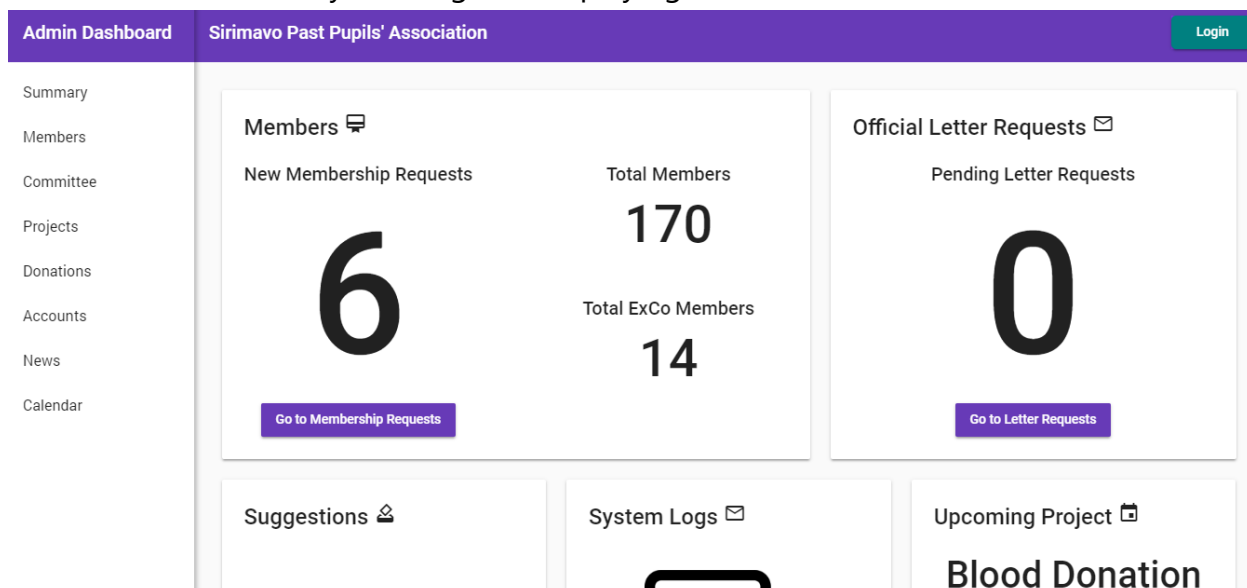
```

Figure 4: summary-view.component.ts

created the above dummy methods in the typescript file of the summary-view component so they can pass the data from the database to its view. There are 7 methods to cater the respective fields in the admin dashboard.

Testing

The numerical values of the summary view cards have bound to the template file using one-way data binding method instead of hardcoding. When the summary view is loaded the values are correctly fetching and displaying from the relevant methods.



3.3 Task 26.1

(Commits: [3b440fa](#), [13371bf](#))

News Screen GUI Design

Estimate Time: 1 Hour

Actual Time: 1 Hour

Actual Time (this sprint): 1 Hour

Description

After thinking and drafting many UIs for the News interface finally I came up with the following wireframe. Since the News should be displayed in the homepage I thought the skeleton of a news consists only with a title and a body text. There is no need of adding lengthier description as a news since it should easily readable. I decided about the news skeleton based on that fact. The reason for adding an expiration date for a news is, usually most of news some with a valid time period. So when adding an expiration date to a news our system can look for the expired news and automatically delete them from the system. That saves the admins' time to removing invalid news. I thought it would be really convenient to add a list which shows existing news in the system, so the admin can easily edit or remove those news items just by clicking on the relevant news.

The wireframe shows a web application titled "PPA Membership Maintenance System". On the left is a sidebar menu with the following items: Summary, Committee, Members, Donations, News (highlighted in purple), Projects, Calendar, and Accounts. The main content area is divided into two sections. The left section is titled "Add A News" and contains three input fields: "News Title", "News" (a larger text area), and "Expiration Date". Below these fields are two buttons: "Cancel" and "Add". The right section is a list of "Sample News" items, including "Sample News 1", "Sample News 2", "Sample News 3", "Sample News 4", and an ellipsis.

3.3 Task 26.2

(Commits: [47b7f1e](#), [928338f](#), [5656de4](#), [58d7a58](#))

News Screen GUI Implementation


Estimate Time: 3 Hours







Actual Time: 2.5 Hours

Actual Time (this sprint): 2.5 Hours

Commits & Build Reports

- [47b7f1e](#)
- [928338f](#)
- [5656de4](#)
- [58d7a58](#)







	Janith Ronaka	47b7f1e <small>M</small>	Merge branch 'master' of https://bitbucket.org/Computing_Projects_SLIIT/2018_sd0...	19 hours ago	
	Janith Ronaka	928338f	PPA-238: Main GUI Draft Designing	19 hours ago	
-					
	Janith Ronaka	58d7a58 <small>M</small>	Merge branch 'master' of https://bitbucket.org/Computing_Projects_SLIIT/2018_sd0...	10 hours ago	
	Janith Ronaka	5656de4	PPA-238: Main GUI Draft Designing	11 hours ago	

1. [47b7f1e - Build Report](#)

#30  Merge branch 'master' of https://bitbucket.org/Computing_Proje...  **Successful** 19 hours ago 1 min 13 sec
Janith Ronaka  [47b7f1e](#)  master

2. [58d7a58 – Build Report](#)

Pipeline	Status	Started	Duration
#36  Merge branch 'master' of https://bitbucket.org/Computing_Proje... Janith Ronaka  58d7a58  master	 Successful	10 hours ago	1 min 19 sec

Description

When implementing the news screen, I added an addition field as "Set an Expiration Date" which was not in the initial wireframe. By adding this field I could make the news item either an expiring news or a non-expiring news. Also I added a delete button which is initially in disabled state, and active only when an existing news item is loaded.

The screenshot shows the 'Add A News' form within the 'Sirimavo Past Pupils' Association' Admin Dashboard. The dashboard has a purple header with 'Admin Dashboard' and 'Sirimavo Past Pupils' Association'. A sidebar on the left lists navigation items: Summary, Members, Committee, Projects, Donations, Accounts, News (highlighted), and Calendar. The main content area features a white card titled 'Add A News'. Inside the card, there is a 'News Title' text input field, a larger 'News Info' text area, and an 'Add An Expiration Date' dropdown menu. At the bottom of the card are three buttons: 'Delete' (disabled), 'Cancel', and 'Add' (active). To the right of the card, there is a list of 'Sample News' items: 'Sample News 1', 'Sample News 2', and 'Sample News 3', each with an edit icon.

Technical Details

I used a material card element to display the news editing fields. I used this because in admin dashboard also I have used the material cards to layout the dashboard items accordingly. Therefore by using a material card in the news section it helps to maintain the consistent appearance throughout the application. I wrapped the input fields within the material input fields as they adds a minimal theme to the elements. I used the following angular material documentation as a reference when designing the news form layout.

- [Angular Material Documentation](#)

Here are the material modules I used for creating the news form screen.

- MatInputModule
- MatFormFieldModule
- MatSelectModule
- MatDatepickerModule
- MatNativeDateModule
- MatSelectModule

```

admin-news.component.html x
1 <div class="container">
2   <div id="content">
3     <div fxLayout="row" fxLayoutGap="5%">
4       <mat-card fxFlex="60">
5         <mat-card-title>
6           <h3>Add A News</h3>
7         </mat-card-title>
8         <mat-card-content>
9           <div fxLayout="column">
10            <mat-form-field appearance="outline">
11              <mat-label>News Title</mat-label>
12              <input matInput placeholder="News Title">
13            </mat-form-field>
14            <mat-form-field appearance="outline">
15              <mat-label>News Info</mat-label>
16              <textarea matInput placeholder="News Info"></textarea>
17            </mat-form-field>
18            <mat-form-field appearance="outline">
19              <mat-label>Add An Expiration Date</mat-label>
20              <mat-select placeholder="Add An Expiration Date" (selectionChange)="checkExDate($event.val
21                <mat-option value="yes">Yes</mat-option>
22                <mat-option value="no">No</mat-option>
23              </mat-select>
24            </mat-form-field>
25            <mat-form-field fxShow="{{visible}}" appearance="outline">
26              <mat-label>Expiration Date</mat-label>
27              <input matInput [min]="minDate" [matDatepicker]="picker" placeholder="Choose a date" disab
28              <mat-datepicker-toggle matSuffix [for]="picker"></mat-datepicker-toggle>
29              <mat-datepicker #picker disabled="false"></mat-datepicker>
30            </mat-form-field>
31            <div fxLayout="row" fxLayoutGap="4%" fxLayoutAlign="end" >
32              <button mat-button fxFlexAlign="start" color="warn" disabled>Delete</button>
33              <button mat-raised-button (click)="onClick()">Cancel</button>

```

Figure 5: admin-news.component.html

```

admin-news.component.css x
1 .example-container {
2   display: flex;
3   flex-direction: column;
4 }
5
6 .example-container > * {
7   width: 100%;
8 }
9
10 #content{
11   padding: 30px 10px;
12 }
13
14 .card{
15   padding: 40px 10px;
16   text-align: center;
17   border-color: blueviolet;
18   margin-left: 20px;
19 }
20
21 mat-nav-list {
22   flex-grow: 1;
23   overflow: auto;
24 }
25
26 textarea{
27   height: 200px;
28 }
29

```

Figure 6: admin-news.component.css

3.4 Task 26.3

(Commits: [0ed5e22](#))

News Screen Database Design


Estimate Time: 1 Hour

Actual Time: 1 Hour

Actual Time (this sprint): 1 Hour

Commits

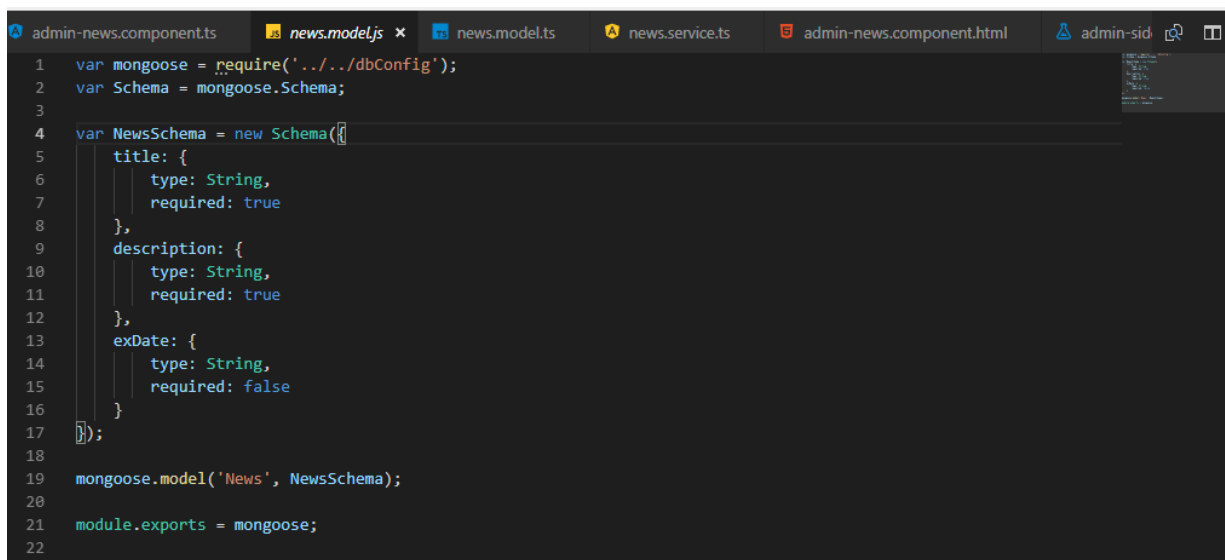
- [0ed5e22](#)

 Janith Ronaka 0ed5e22 PPA-246: Add/Edit News Items database design 26 minutes ago

Description

When designing a collection to save the news information I did it considering that this should easily access by both homepage and the admin news component. Since the client need to have only very short notices type news I only added the most mandatory fields.

Technical Details



```
1 var mongoose = require('../dbConfig');
2 var Schema = mongoose.Schema;
3
4 var NewsSchema = new Schema({
5   title: {
6     type: String,
7     required: true
8   },
9   description: {
10    type: String,
11    required: true
12  },
13  exDate: {
14    type: String,
15    required: false
16  }
17 });
18
19 mongoose.model('News', NewsSchema);
20
21 module.exports = mongoose;
22
```

Figure 7: news.model.js

	_id ObjectId	title String	description String	exDate String	__v Int32
1	5b882f262ac7bc3440d443d2	"N1"	"Sample news body Sample news body Sa	"Fri Aug 31 2018 00:00:00 GMT+0530 (Ir	0
2	5b882f2f2ac7bc3440d443d3	"N2"	"Sample news body Sample news body Sa	"Fri Aug 31 2018 00:00:00 GMT+0530 (Ir	0

Figure 8: News Collection in MongoDB Compass

3.5 Task 26.4

(Commits: [ff5f5d7](#))

News Screen Validation Methods

Estimate Time: 1 Hours

Actual Time: 2 Hours

Actual Time (this sprint): 2 Hours

Commits & Build Reports

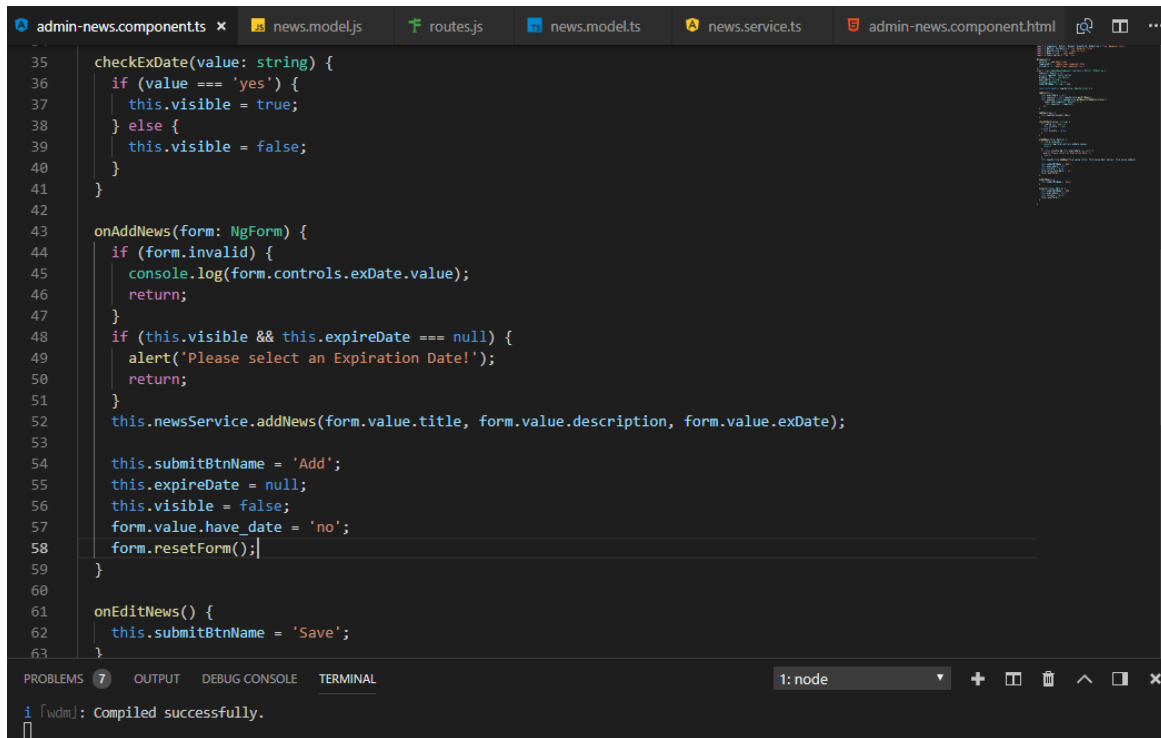
- [ff5f5d7](#)

Janith Ronaka ff5f5d7 PPA-247: Add/Edit News Items Middle Tier 34 minutes ago

Description

I did the form validations when a news item is added. First it checks the default form validations and then it validate the expiry date field manually when submitting. If the validations are passed successfully it calls for the addNews method in the NewsService.

Technical Details



```
admin-news.component.ts x news.model.js routes.js news.model.ts news.service.ts admin-news.component.html ...
35 checkExDate(value: string) {
36   if (value === 'yes') {
37     this.visible = true;
38   } else {
39     this.visible = false;
40   }
41 }
42
43 onAddNews(form: NgForm) {
44   if (form.invalid) {
45     console.log(form.controls.exDate.value);
46     return;
47   }
48   if (this.visible && this.expireDate === null) {
49     alert('Please select an Expiration Date!');
50     return;
51   }
52   this.newsService.addNews(form.value.title, form.value.description, form.value.exDate);
53
54   this.submitBtnName = 'Add';
55   this.expireDate = null;
56   this.visible = false;
57   form.value.have_date = 'no';
58   form.resetForm();
59 }
60
61 onEditNews() {
62   this.submitBtnName = 'Save';
63 }
```

PROBLEMS 7 OUTPUT DEBUG CONSOLE TERMINAL 1: node

i [vdm]: Compiled successfully.

Figure 9: admin-news.component.ts

Testing

1. Test Case: Submitting an empty form

Sirimavo Past Pupils' Association

Add A News

News Title *

News Title Cannot be Empty

News Info *

News Info Cannot be Empty

Add An Expiration Date

No

Delete Cancel Add

Expected Behavior: Field validation should raise the errors and mark the fields with validation errors.

Test Status: Passed

2. Test Case: Submitting a news without a expiry date

localhost:4200/admin-news

Dashboard Sirimavo Past Pupils' Association

localhost:4200 says
Please select an Expiration Date!

OK

Add A News

News Title *

News 3

Expected Behavior: Field validation should raise an alert box with the message "Please select an Expiration Date!"

Test Status: Passed

3.5 Task 26.5, 26.6, 26.7

- (Commits: [2c31592](#), [92dde1e](#))

News Screen Insert Methods

News Screen Update Methods

News Screen Delete Methods




Estimate Time: 6 Hours

Actual Time: 6 Hours

Actual Time (this sprint): 6 Hours

Commits & Build Reports

- [2c31592](#)
- [92dde1e](#)

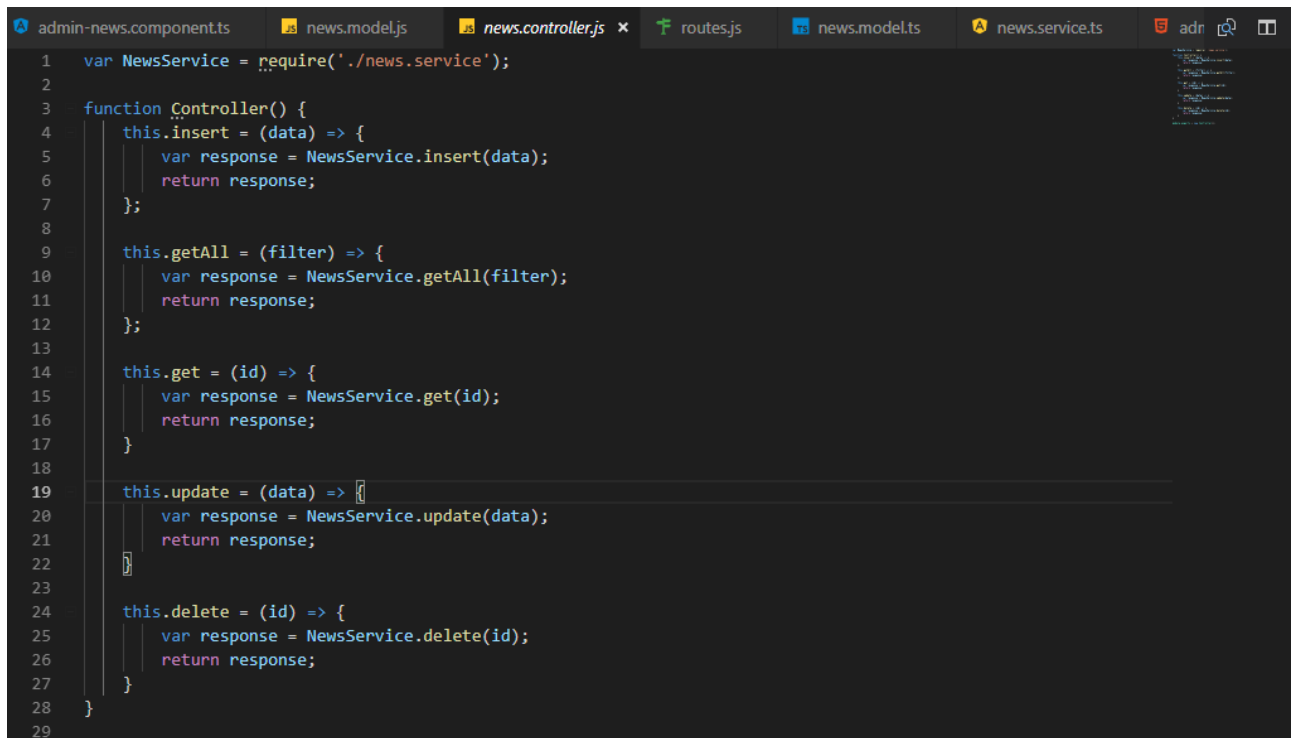
Pipeline	Status	Started	Duration
#45  Merge branch 'master' of https://bitbucket.org/Computing_Proje... Janith Ronaka  92dde1e  master	 Successful	27 minutes ago	1 min 14 sec

1. [92dde1e – Build Report](#)

Description

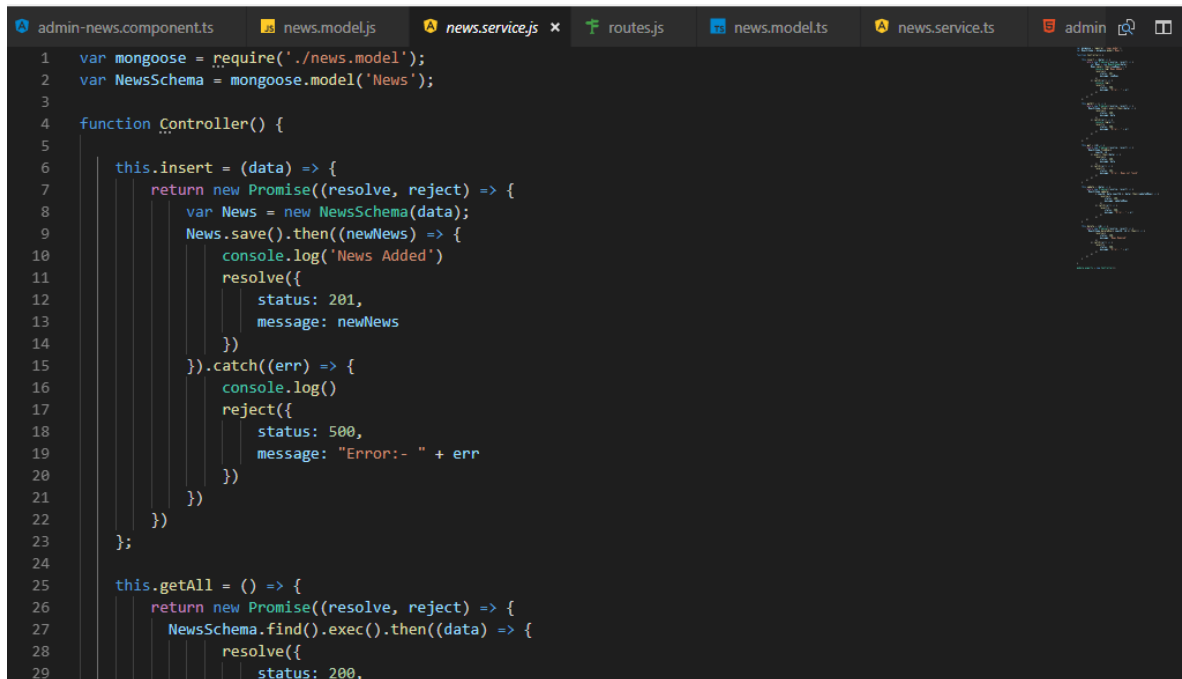
I have implement an insert method in news.service.js file to handle the direct db transactions for insert calls. The insert method in the news.controller.js file is a wrapper method of the insert method in the news.service.js. All the insert method calls from the components directs to the db through this wrapper insert method. In the news.route.js file I configured the accessible routes for all the insert, update, delete and get methods.

Technical Details



```
1 var NewsService = require('./news.service');
2
3 function Controller() {
4   this.insert = (data) => {
5     var response = NewsService.insert(data);
6     return response;
7   };
8
9   this.getAll = (filter) => {
10    var response = NewsService.getAll(filter);
11    return response;
12  };
13
14  this.get = (id) => {
15    var response = NewsService.get(id);
16    return response;
17  }
18
19  this.update = (data) => {
20    var response = NewsService.update(data);
21    return response;
22  }
23
24  this.delete = (id) => {
25    var response = NewsService.delete(id);
26    return response;
27  }
28 }
29
```

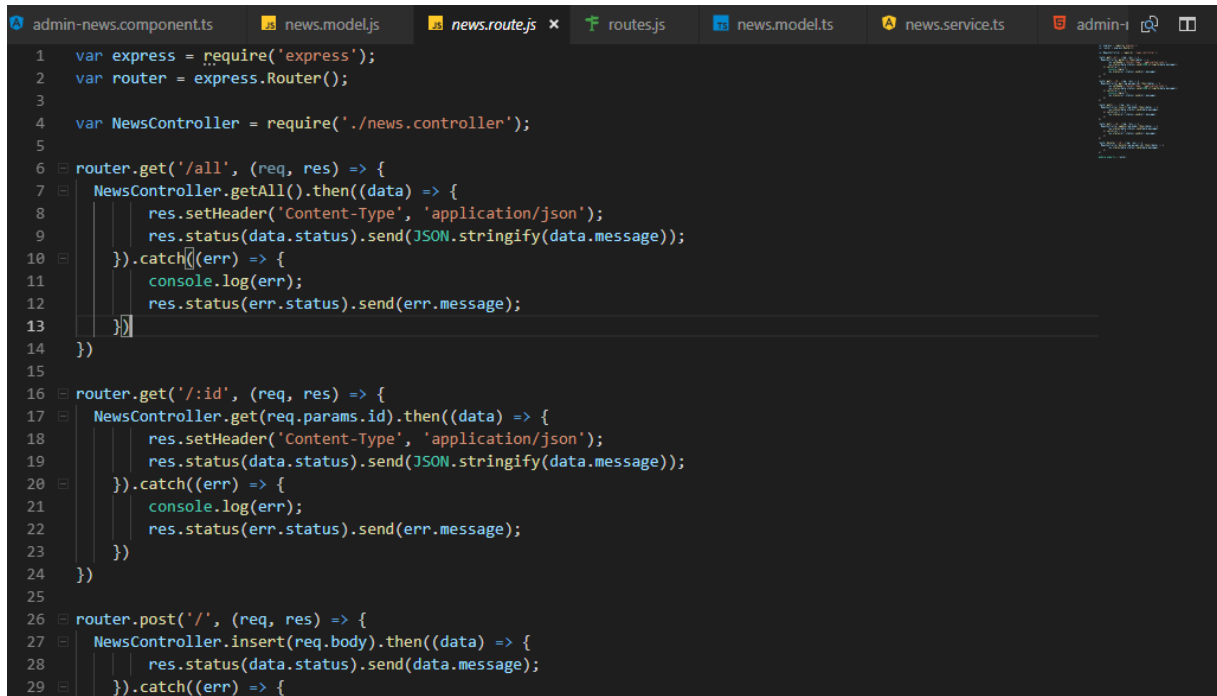
Figure 10: news.controller.js



```
1 var mongoose = require('./news.model');
2 var NewsSchema = mongoose.model('News');
3
4 function Controller() {
5
6   this.insert = (data) => {
7     return new Promise((resolve, reject) => {
8       var News = new NewsSchema(data);
9       News.save().then((newNews) => {
10         console.log('News Added')
11         resolve({
12           status: 201,
13           message: newNews
14         })
15       }).catch((err) => {
16         console.log()
17         reject({
18           status: 500,
19           message: "Error:- " + err
20         })
21       })
22     })
23   };
24
25   this.getAll = () => {
26     return new Promise((resolve, reject) => {
27       NewsSchema.find().exec().then((data) => {
28         resolve({
29           status: 200,
```

Figure 11: news.service.js

I have imported the news.service.js file in news.controller because it carries the direct db transaction methods of the wrapper methods in this file. I access those methods by the NewsService variable which is created using the news.service.



```
1 var express = require('express');
2 var router = express.Router();
3
4 var NewsController = require('./news.controller');
5
6 router.get('/all', (req, res) => {
7   NewsController.getAll().then((data) => {
8     res.setHeader('Content-Type', 'application/json');
9     res.status(data.status).send(JSON.stringify(data.message));
10   }).catch((err) => {
11     console.log(err);
12     res.status(err.status).send(err.message);
13   })
14 })
15
16 router.get('/:id', (req, res) => {
17   NewsController.get(req.params.id).then((data) => {
18     res.setHeader('Content-Type', 'application/json');
19     res.status(data.status).send(JSON.stringify(data.message));
20   }).catch((err) => {
21     console.log(err);
22     res.status(err.status).send(err.message);
23   })
24 })
25
26 router.post('/', (req, res) => {
27   NewsController.insert(req.body).then((data) => {
28     res.status(data.status).send(data.message);
29   }).catch((err) => {
```

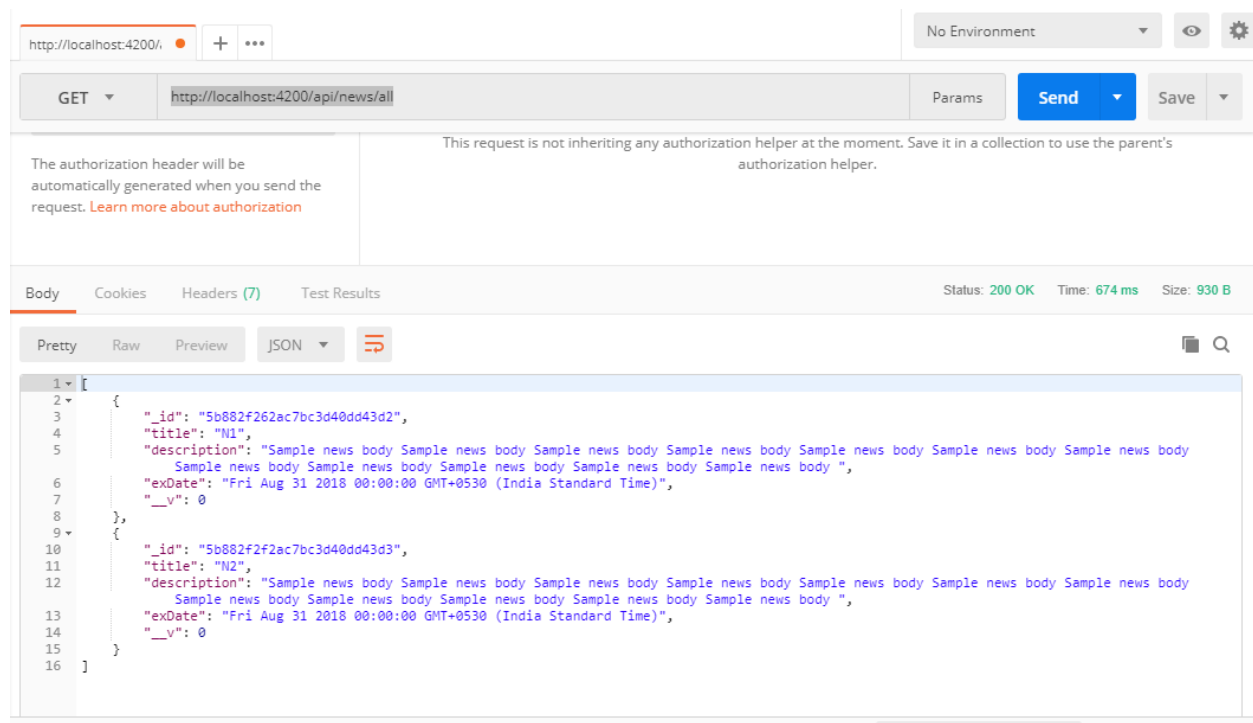
Figure 12: news.route.js

In the `news.route.js` file configured separate routing paths for each method in news controller. In each of methods I have setHeader content type of the response objects' as `application.json`. All the responses are sent back to the components as json strings.

Testing

All the following tests are done using the postman tool

1. Test Case: Get all news items



Expected Behavior: Should return all the news items in the db with the correct status code.

Test Status: Passed

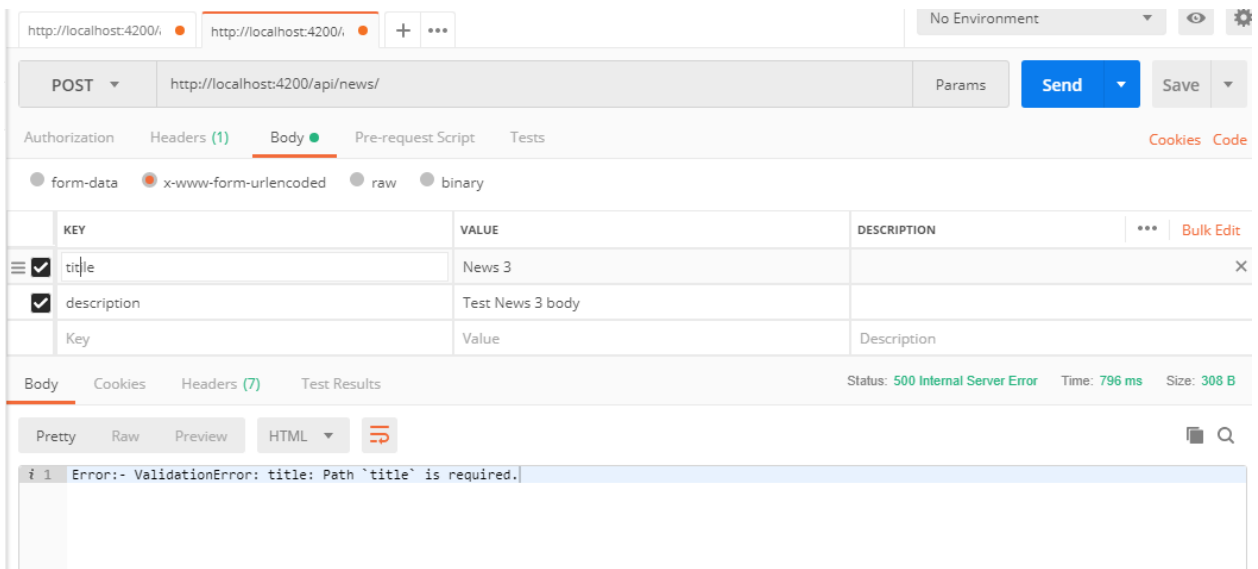
2. Test Case: Get single news item



Expected Behavior: Should return the correct news item with the correct status code.

Test Status: Passed

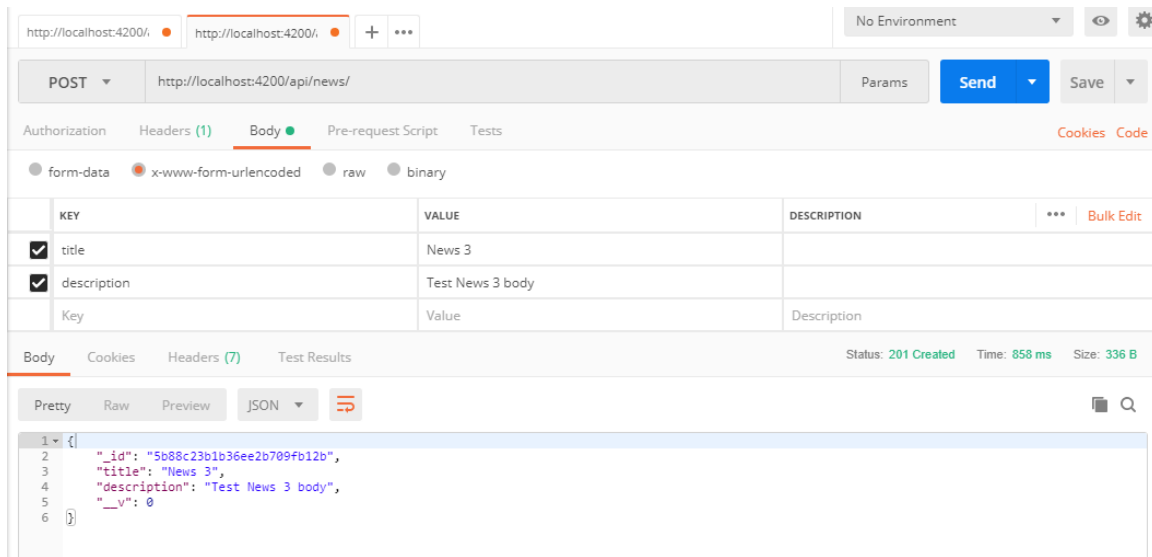
3. Test Case: Give an error when a required field is empty



Expected Behavior: When a required field is not given in the post request, the validation errors raised correctly.

Test Status: Passed

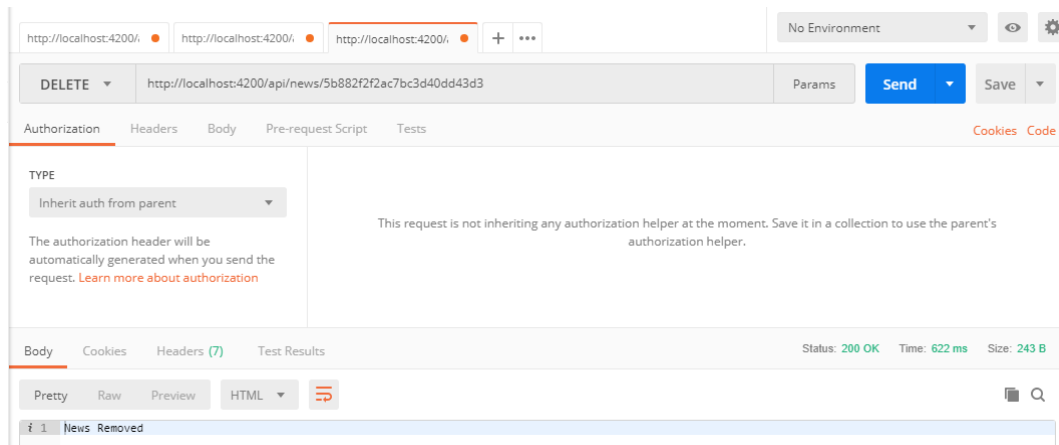
4. Test Case: Insert a new item to the db



Expected Behavior: When all the required fields are given new item should be inserted and return the correct status code.

Test Status: Passed

5. Test Case: Deleting an existing news item



Expected Behavior: Should delete the news item and return the response message and status code correctly.

Test Status: Passed

6. Test Case: Updating an existing news item

The screenshot shows a REST client interface with a PUT request to `http://localhost:4200/api/news/5b882f262ac7bc3d40dd43d2`. The request body is a JSON object with the following fields:

Key	Value	Description
<input checked="" type="checkbox"/> title	News 1	
<input checked="" type="checkbox"/> description	Test News 1 body	
<input checked="" type="checkbox"/> _id	5b882f262ac7bc3d40dd43d2	

The response is a JSON object with the following fields:

```
1 {
2   "n": 1,
3   "nModified": 1,
4   "opTime": {
5     "ts": "6595742819330031622",
6     "t": 7
7   },
8   "electionId": "7ffffffff00000000000000007",
9   "ok": 1,
10  "operationTime": "6595742819330031622",
11  "$clusterTime": {
12    "clusterTime": "6595742819330031622",
13    "signature": {
14      "hash": "pSh0BfLGD0Iwn388elCLi15zsq0k=",
15      "keyId": "6563302068054917121"
16    }
17  }
18 }
```

Expected Behavior: Should update the news item and return the updated object.

Test Status: Passed

3.7 Task 8

Design Meetings

Estimate Time: 2 Hours

Actual Time: 3 Hours

Actual Time (this sprint): 3 Hours

Description

Please find the design meeting minutes from the following link.

Bitbucket Link:

https://bitbucket.org/Computing_Projects_SLIT/2018_sd07/commits/6b5d0d63a8948eb0cc9dbbcff71c9b105ed10e29#chg-Documents/Design%20Meetings/Design%20Meeting%20-%20Sprint%202.docx

3.8 Task 6.2

Planning the user stories for the sprint 3

Estimate Time:	1 Hour
Actual Time:	1 Hour
Actual Time (this sprint):	1 Hour

Description

Please find the sprint 3 planning meeting minutes from the following link.

Bitbucket Link:

https://bitbucket.org/Computing_Projects_SLIT/2018_sd07/src/master/Documents/Sprint%20Documents/Sprint%203/Sprint%203%20Planning%20Minutes.pdf

3.9 Task 7.2

Sprint 2 sprint retrospective

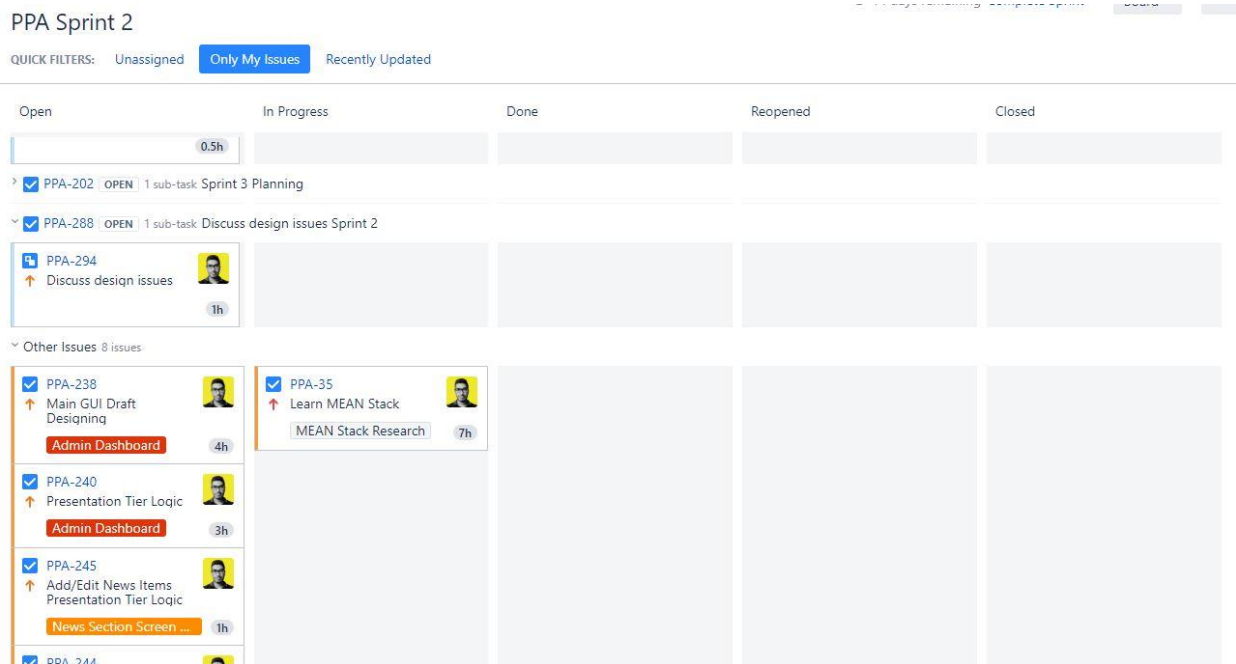
Estimate Time:	0.5 Hours
Actual Time:	0.5 Hours
Actual Time (this sprint):	0.5 Hours

Please refer the following Sprint Retrospective section for more details.

4 Development Methodology

4.1 Minutes

We use Jira as the project management tool for our project. At the one of our group members setup the Jira in Amazon cloud. There after we prepared a basic backlog for the sprint 1 tasks and started a new sprint for this sprint by adding those tasks. We set estimated times and assigned the tasks to relevant members. Following is a screen shot of the sprint board which shows some of the tasks that were allocated for me at the beginning of this sprint.



We had 12 standup meetings and 1 design meeting within this sprint and the minutes of those meetings can be found from the following link.

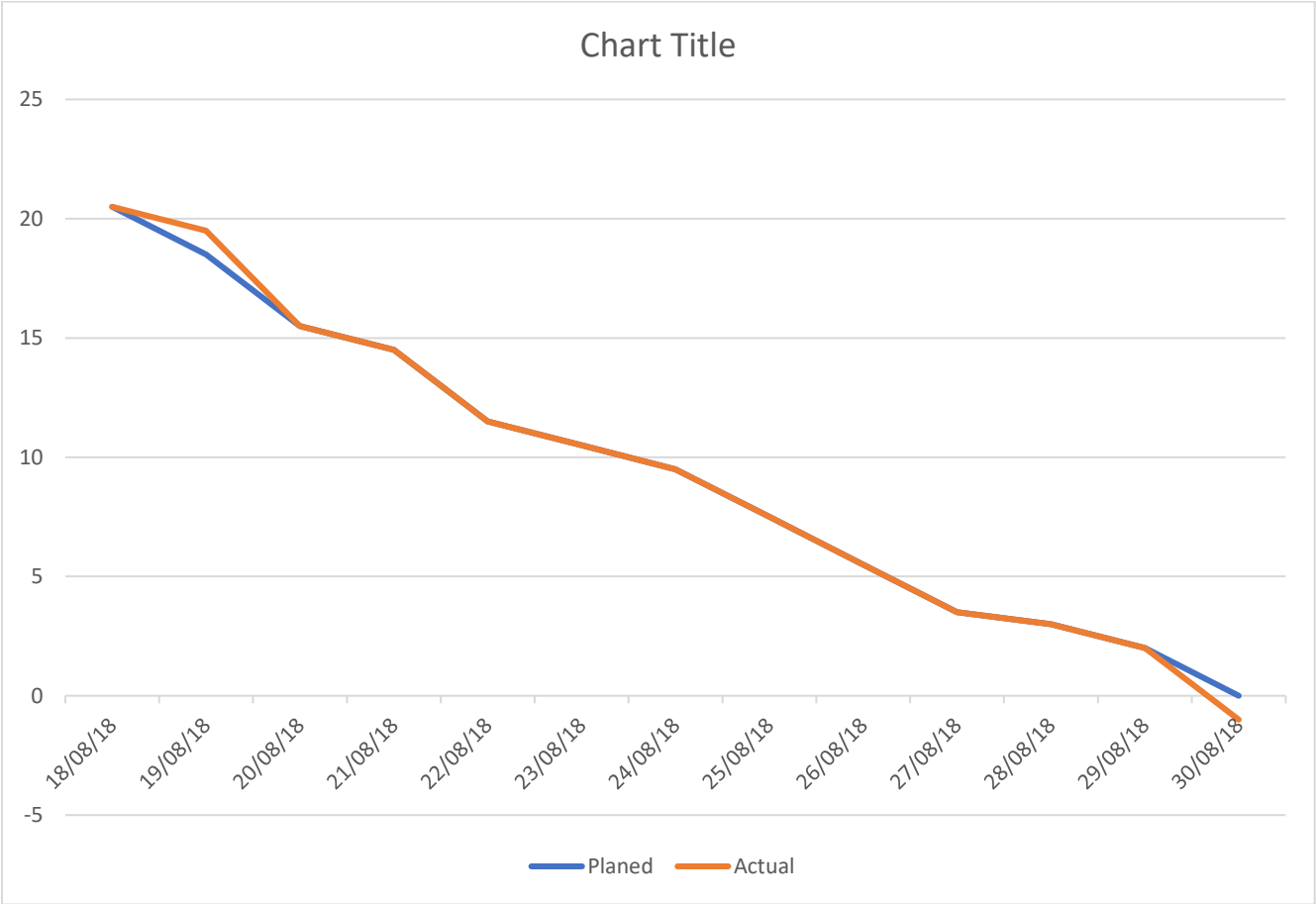
Standup Meeting Minutes:

https://bitbucket.org/Computing_Projects_SLIIT/2018_sd07/src/master/Documents/Standup%20Meeting%20Minutes/Sprint%202/Standup%20Meeting%20-%20Sprint%202.pdf

Design Meeting Minutes:

https://bitbucket.org/Computing_Projects_SLIIT/2018_sd07/commits/6b5d0d63a8948eb0cc9dbb6c71c9b105ed10e29#chg-Documents/Design%20Meetings/Design%20Meeting%20-%20Sprint%202.docx

4.2 Burndown Chart



4.3 Sprint Retrospective (Task 7.2)

Estimate Time: 0.5 Hours

Actual Time: 0.5 Hours

Actual Time (this sprint): 0.5 Hours

Description

In this sprint we started our developments. Since all of us have not a much exposure in MEAN stack domain, there were quite few learning stuff during the developments. We hope that the exponential growth of the learning curve will be come to a more flatten phase in the future sprints as we get familiar with the language and frameworks. As development goes on the heroku build which runs builds after every commit in the online repo was really helpful to detect the errors that were rose when merging. That was easy for us to fix the relevant issue, since that tool points out which commit cause to the error. Some of the initial tasks of the sprint were taken a bit additional time than allocated due to the learning stuff. Apart from that all the planned tasks were went well as expected. We had about 4 standup meetings time to time, to discuss the current status of the each ones allocated tasks. We always tried to keep the look and feel of the UIs by discussing and reviewing each other's work during these meetings. We also used a common flow of development to achieve the tier based architecture which was decided in the design meeting. Jira tool was quite helpful to track each of us progress during the sprint. Its real time sprint burndown chart was came quite handy when estimating and planning the remaining workload of the sprint, in both perspective as a team as an individual. Please refer the following two snapshots of the burndown chart (in team perspective and individual perspective) which were taken at some point during the sprint 2 for more clearance.

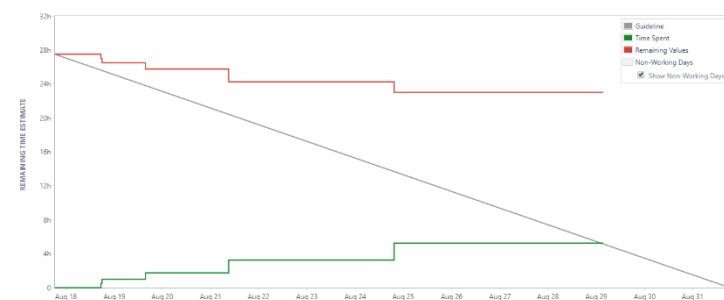


Figure 13: Individual Burndown Chart

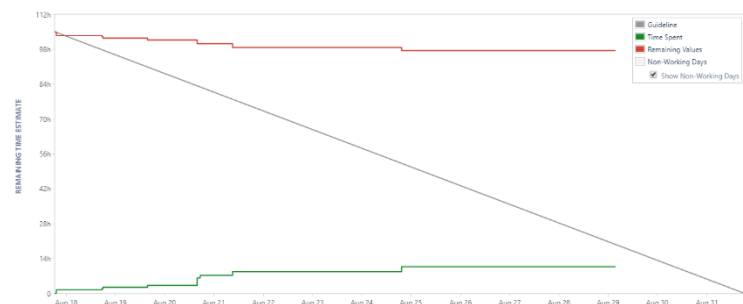


Figure 14: Team Burndown Chart

4.4 Time Management

Task ID	Task	Task Status	Estimated Times (h)	Actual Times (h)
Task 13.1	Admin Dashboard - Main GUI Designing	Completed	2	1
Task 13.2	Admin Dashboard - Main GUI Implementation	Completed	3	4
Task 26.1	News Screen GUI Design	Completed	1	1
Task 26.2	News Screen GUI Implementation	Completed	3	3
Task 26.3	News Screen Database Design	Completed	1	1
Task 26.4	News Screen Validation Methods	Completed	1	2
Task 26.5	News Screen Insert Methods	Completed	2	2
Task 26.6	News Screen Update Methods	Completed	2	2
Task 26.7	News Screen Delete Methods	Completed	2	2
Task 7.2	Sprint 2 Retrospective	Completed	0.5	0.5
Task 6.2	Planning the user stories for the sprint 3	Completed	1	1
Task 8	Design Meetings	In Progress	2	3
Total Time			20.5	22.5