Mini Project Report

on

BRIDGE

Student Self Monitoring System

submitted by

Abin Simon (12150005)
Aayisha A A (12150076)
Abhai Kollara (12150002)

In partial fulfilment of the requirements for the award of degree of Bachelor of Technology in Computer Science and Engineering.

DIVISION OF COMPUTER ENGINEERING
SCHOOL OF ENGINEERING
COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY

MARCH 2017

DIVISION OF COMPUTER ENGINEERING
SCHOOL OF ENGINEERING

COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Certificate

Certified that this is a bonafide record of the Minor Project titled

BRIDGE
Student Self Monitoring System

done by

Abin Simon (12150005)
Aayisha A A (12150076)
Abhai Kollara (12150002)

of VI Semester, Computer Science and Engineering in the year 2017 in partial fulfillment requirements for the award of degree of Bachelor of Technology in Computer Science and Engineering of Cochin University of Science and Technology.

Ancy Zachariah	Pramod Pavithran / Damodaran.V	Anu M.
Head of Division	Project Coordinator	Project Guide

ACKNOWLEDGEMENT

We take this opportunity to express our profound gratitude and deep regards to our guide **Mrs Anu M.** ,for providing us with the right guidance and advice at the crucial junctures and for her constant encouragement throughout the course of this project. We are highly indebted to **Associate Prof. Pramod Pavithran** and
Associate Prof. Damodaran.V , Division of Computer Science our batch coordinator for her constant supervision and support for completing the project. We extend our sincere thanks to our respected Head of the department, **Miss Ancy Zachariah** ,Head of Division,Division of Computer Science and all other faculty members of for sharing their valuable time and knowledge with us. We thank God, the almighty for blessing us with his grace and taking our endeavour to a successful culmination. Lastly we would like to thank my friends and family for their constant encouragement without which this project would not have been possible.

Thanking you,

Abin Simon
Aayisha A A
Abhai Kollara

ABSTRACT

We are living in an age where the art of technology and the art of teaching has come a long way. With bridge we intent to interlace those by helping to bridge the communication gap between teachers and students. Currently, even though teachers use technology to reach out to students the solutions are more or less ad-hoc. The communication is spread over different channels, sometimes a phone call, sometimes a text message, and sometimes nothing at all. There lacks an efficient way for teachers to communicate to students directly as a group or individually. Also there is no easy way for the students and teachers to keep track of the tasks they have in hand. It is at times really hard for teachers to reach students most of time and they end up relying on the class representative to deliver the message which in itself is not bad, but is less efficient.

Bridge intents to solve all the above stated problems by implementing a application using which the teachers and students can communicate seamlessly with each other, keep track of everything that they have to do, get replies asap and just about anything that helps make the student teacher communication much more effective and easy.

Contents

List of Figures

Chapter 1

Introduction

The most important aspect of a academic studies for a student is the involvement of the teachers in the matters of the students. It is also very vital for the student to stay updated about what is happening in the college. With this project we aim to do just that. We are creating a platform that can help the students to easily stay up to date on what is happening in their academic matters. They have an easy and intuitive way to see what is in their calendar.

Our platform also provides an effective way for the student to track his daily activities at the academic institute by giving then a simple and efficient platform to take notes and track their attendance. This, we think is a great tool that will help the student of the academic institution to be able to track his time at the institution.

The platform with its ability to take down notes for the classes they are attending in the university, in a platform that has their timetable and other data integrated along with it is a huge bonus for the student as it gives them a seamless way to take down notes for a specific period and let them filter it and get all the dada beautifully presented to them which is a huge bonus for a student.

Chapter 2

System Analysis

2.1 Existing system

The present system is ineffective in maintaining a student centric model. It mainly consisted of web applications and very few mobile applications. The traditional way of maintaining record is time consuming, not easily accessible ,requires a computer, less user friendly, and laborious.

Apart from these all the system we have currently are mostly teacher centric which leads to a hard time for the students to navigate and find what they need.

Here are some disadvantages of existing system:

- Time consuming
- Not easily accessible
- Teacher centric
- Less student friendly
- Laborious

2.2 Proposed system

With our solution we are aiming to provide a simple and intuitive interface for the student. We are also looking forward to provide a student centric model in which the student will able to track his classes at the university.

Once the user get registered through their google sign in (which makes the signin and verification process efficient and easy). The user will be provided with a popup menu in which they could select their respective division head and to which course they are enrolled to.after the completion of sign in process, we will have a wide range of options which include attendance tracking system, notes adding, which could include lecture videos, images and much more feature which make learning easier, it tend to have an added feature of upcoming events which help to clear the backlogs and make the work as timid and as clean as possible

- This system is developed in such a way that even a naive user can also operate the system easily.
- This system is also secure as the database is managed only by the administrator of the system.
- It has an error free verification mechanism
- Easy way of accessing records and tracking attendance

Chapter 3

System Study

3.1 Software Requirement Specification

3.1.1 Introduction

Purpose

The purpose of this document is to present a detailed description of the Web Publishing System. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate and how the system will react to external stimuli.

Intended Audience and Reading Suggestions

This document is intended for the understanding of complete functionalities that will be provided by the software. Therefore it may be used by anyone who wishes to further develop this software or anyone who intends to understand the full capabilities of the system.

Scope of the Project

The software system will be a Student Self Monitoring System intended for the use by students of a college. It is designed so as to help students keep track of their academic activities in college. We include attendance tracking, tasks tracking, notes feature etc for increasing the productivity of students. The attendence tracking mechanism is an extremely useful tool. The notes features allow students to save notes for each period of a day. The feature also supports markdown text which allows the student to go beyond plaintext for their notes.

It is also intended to to minimize the difficulties of communication between students and college authorites which would otherwise have to be performed manually by phone calls or class representatives. More specifically the system allows teachers to notify the students about upcoming tasks/events increasing the efficiency of communication.

The system also contains a relational database containing Students, Teacher, Departments, Classes, Subjects, Notes, Events.

References

IEEE. IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications. IEEE Computer Society, 1998.

Django Official Documentation

Overview of Document

The next chapter, the Overall Description section, of this document gives an overview of the functionality of the product. It describes the informal requirements and is used to establish a context for the technical requirements specification in the next chapter.

The third chapter, Requirements Specification section, of this document is written primarily for the developers and describes in technical terms the details of the functionality of the product.

Both sections of the document describe the same software product in its entirety, but are intended for different audiences and thus use different language.

3.1.2 Overall Description

Product Perspective

The software is designed to be a stand-alone web application that is designed to provide a different approach to current learning management systems like Moodle.

Product Functions

The primary functions of the software are:

- Let the students store notes of every period
- Let the students store and track their attendence of every period
- Let teachers/admins create and assign events to classes
- Let students view upcoming classes

User Classes

The Student Self Monitoring System has two main actors, the user (Student) and the admin (Teachers/other authorities). The role of the admin is to provide the system with sufficient data about each classes, subjects, departments and teachers involved. The admin has near complete access to the database. It is also the role of the admin(teacher) to add upcoming events/tasks to the software system.

Operating Environment

The software is designed to be platform independent and to work on any major web browser that supports Javascript. The software is also designed resposively so as to run on mobile web browsers.

Design and Implementation Constraints

The major design constraint was to implement the software to run on mobile platforms. The UI should be designed responsively to accomodate mobile view

Assumptions and Dependencies

The software requires a server that supports running Django 1.11 on Python 2.7

3.1.3 External Interface Requirements

User Interface

The UI must be responsive and must view correctly on mobile views. The UI is intended to be platform independent ie it should be the same irrespective of the OS and web browser. A material design is applied thorought the application

Hardware Interfaces

From client side, devices with at least 512 MB of RAM is recommended along with a low latency high bandwidth internet connectivity. This is applicable for both desktop and mobile systems.

Software Interfaces

The software communicates with a MySQL database through the Django interface. Basic built-in python libraries are assumed available.

The software also accesses the Google sign-in API for secure login.

3.1.4 System Features

This section outlines the use cases for each of the actors seperately; the user and the admin.

Admin Use Case

- The admin must be able to add/remove data to/from the following entities :
  - Departments
  - Teachers
  - Subjects
  - Classes
- The admin must also be able to specify the relation between teacher, subject and classes.
- The admin must be able to add new events to the upcoming event list for the users to view

User Use Case

- Login – User must be able to login using Google Accounts
- User must be allowed to pick a class of his choice
- Attendance Tracking

  - The user must be able to view the current attendance
  - The user must be update the attendace on a daily basis

- Notes
  - User must be able to save notes for every period in a day
  - Markdown text should be supported
  - Images should be supported
  - User must be able to view all the notes saved as of yet

- Upcoming Events
  - User must be able to view upcoming events that are added by admins
  - User must have a calender view of upcoming events

3.1.5 Other Non-Functional Requirements

Safety Requirements

Logins must be completely secure and only admins must be allowed to access administrator interfaces.

3.2 Hardware and Software Requirements

3.2.1 Hardware specification

- RAM: Recommended 512MB or above
- Storage: 10 GB or above
- Connectivity: Low latency internet with high bandwidth

3.2.2 Software specification

- OS: Windows, OSX, or Linux (or any unix system)
- Env: Python 2.x
- Database: MySQL
- Web browser (preferable Chrome or Firefox)

Chapter 4

System Design

4.1 Data Flow Diagrams

The dfd 4.1 below show in the higer level data flow in the project

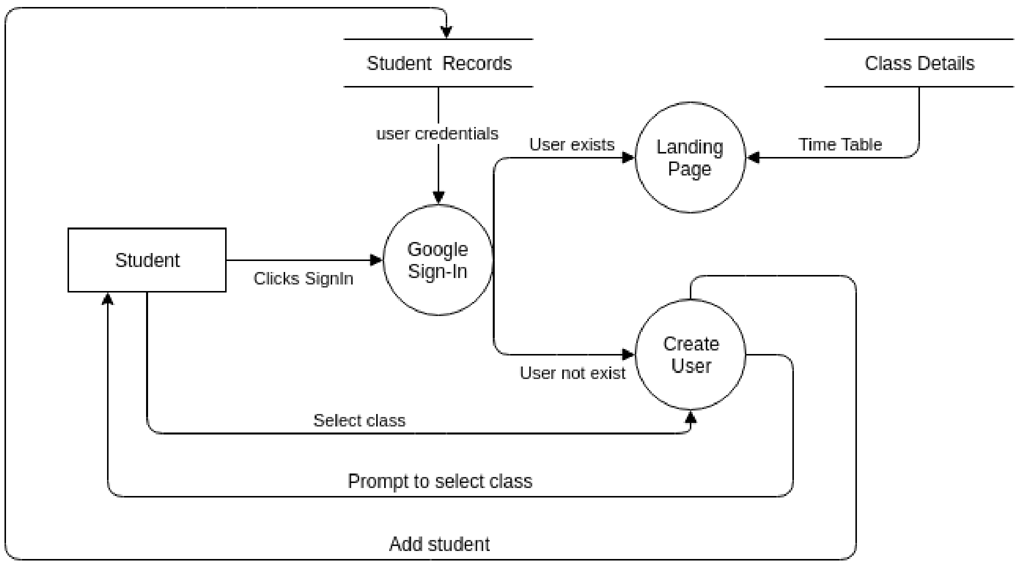


Figure 4.1: First level dfd

This dfd 4.2 shows how the project handles data within

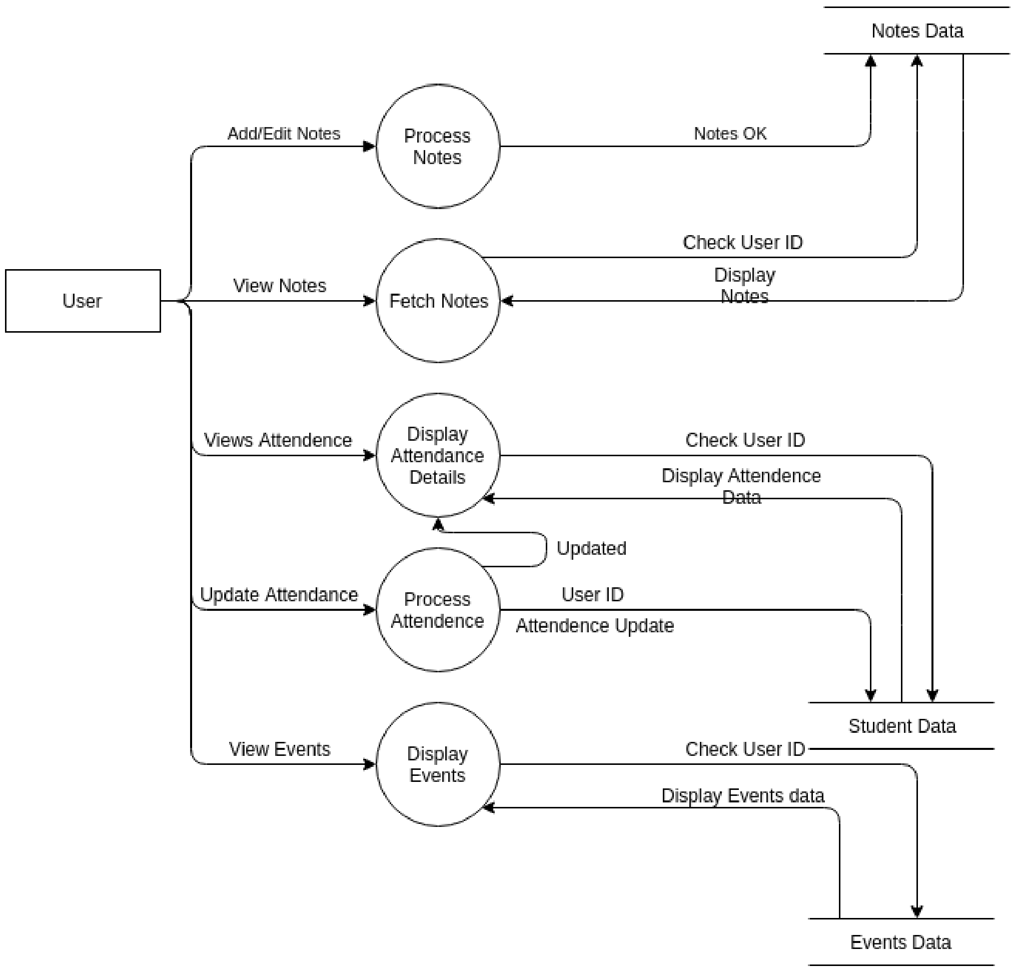


Figure 4.2: Inner dfd

4.2 Modular Design

4.2.1 Login

This module handles the user authentication and login. The component is dependent on Google login for its authentication, as we leverage Google signin module to make the tradition of the user into our app much more easier and faster.

Initially when the user logs into the application is when we create an account for them and register them. When the user logs into the application, we get the user to login to their Google account and we can use the Google API top get the user details like user-id, name, email, profile-image etc. This makes it easer for the user as do not have to manually add in their name or profile image. Only thing they will have to manually set is which class they are studying in.

Now from the next time the user logs in, we will directly log them into their account ready to go.

4.2.2 Landing Page

The landing page is the most important and most viewed part of the whole application. It encompasses details a user would probably need at that time like upcoming events, which data about the class that is currently going on. It also lets them take down notes which they can later view using the notes module.

For each subject of the day we provide the name of the teacher, time of the class, etc which helps the student to plan for their class. The place provided to jot down notes for the subject also helps them to take down notes during the classes which will be really useful for them for later reference.

4.2.3 Notes

Notes module is aimed at providing an interface for the student(user) to view all the notes they have taken in the class and go through them. It is a very useful and powerful utility at the end of a semester as it will let them view all the notes they have in one place and go through them quickly and efficiently.

You can also use the notes view to filter your notes based on the date or the subject which is a really simple but powerful way to summarize a whole semester.

4.2.4 Attendance

One another very important module is the Attendance module which lets the student track their attendance for each subject they have. The student on attending or net attending a specific subject class can update their attendance using the attendance module. All the user data is saved in real-time with the backed and they will be able to check their any time.

The per subject nature of the attendance module also helps them to know which subjects they have missed the most and concentrate on them individually.

4.2.5 Calendar

The Calendar module is used to view the overview of all the events. It shows you all the events and submissions you have for the future. While the upcoming events module only shows you the close and upcoming events, in the calendar module we can see all the upcoming events in the future.

It has a normal calender like intercase with events listed under the specific dates which makes it much more intuitive and easy.

4.3 Input Output design

4.3.1 Input design

The input part of the project includes ways for the user to add data like attendance and notes to the system. It also provides an interface for teacher to add upcoming events for specific classes.

The user can log into our system using their Google account and then start adding up notes and marking their attendance.

4.3.2 Output design

The output part of the project is the more important part as the main goal of the project is to provide the user with information about the education institution actives at their fingertips.

Without any effort the user can view upcoming events and todays day in a glance. They main page also gives them a heads up on the upcoming events in the near future which lets them act wisely.

The user can also view all the notes they have taken down during the year, check their attendance.

Chapter 5

System Implementation

5.1 Sample Code

5.1.1 Sample view handlers

A view handler is a methood which is a specification in django which handles a http request and makes the response.

```python
def signin(request):
    if request.method == "POST":
        data = json.loads(request.POST.get('data'))
        sid = data['id']
        if Student.objects.filter(SID=sid).exists():
            return HttpResponse(json.dumps({'exists':\
                    True}))
        else:
            classes = [name.class_name for name in \
                    Class.objects.all()]
            data = json.dumps({'exists': False, \
                    'options': classes})
            return HttpResponse(json.dumps(data), \
                    content_type="application/json")

def get_attendence(request, user_id):
    user = Student.objects.get(SID=user_id)
    attendence_data = user.get_attendence_data()

    print attendence_data
    if len(attendence_data) is 0:
        # quick patch ( issue only if new user already
                # created)
        daytable = user.current_class.get_tt()
        subs = {}
        for key in daytable:
            daysubs = daytable[key].split(',')
            for item in daysubs:
                # if item not in subs:  ( its a dict, lol )
                subs[item] = {'total':0, 'attended':0 }
        attendence_data = subs

    return_data = {}
    return_data['data'] = attendence_data
    return HttpResponse(json.dumps(return_data), \
            content_type="application/json")

def update_attendence(request):
    if request.method == "POST":
        data =  json.loads(request.POST.get('data'))
        user_id = data['id']
        attendence = json.dumps(data['attendance'])

        user = Student.objects.get(SID=user_id)
        user.attendence = attendence
        user.save()
    return_data = { 'status':'OK' }
    return HttpResponse(json.dumps(return_data), \
            content_type="application/json")
```

5.1.2 Url defenitions

These define the endpoints fot the backend api

```python
from django.conf.urls import url
from . import views

urlpatterns = [
    url(r'^$', views.index, name="index"),
    url(r'^signin/$', views.signin, name='signin'),
    url(r'^create_user/(?P<user_id>[0-9]+)/(?P<user_class>\
            ([a-z])+)$', views.create_new_user, \
            name='new_user'),
    url(r'^timetable/(?P<user_id>[0-9]+)$', \
            views.get_timetable, name='timetable'),
    url(r'^notes/(?P<user_id>[0-9]+)$', \
            views.get_notes, name='notes'),
    url(r'^subject_data/(?P<user_id>[0-9]+)$', \
            views.get_sub_data, name='subject_data'),
    url(r'^events/(?P<user_id>[0-9]+)$', \
            views.get_events_dummy, name='events'),
    url(r'^track_data/$', \
            views.get_track_data, name='events'),
    url(r'^calendar/(?P<user_id>[0-9]+)$', \
            views.get_cal_data_dummy, name='events'),
    url(r'^subject_attendence/(?P<user_id>[0-9]+)$', \
            views.get_attendence, name='get_attendence'),
    url(r'^create_user/$', \
            views.create_new_user, name='new_user'),
    url(r'^update_attendence/$', \
            views.update_attendence,\
            name='update_attendence'),
    url(r'^set_track_data/$', \
            views.set_track_data, name='set_track_data'),
]
```

5.1.3 Model defenitions

These contains the database models for the project

```
 class Subject(models.Model):
    subject_code = models.CharField(\
            verbose_name="Subject Code", max_length=6,\
            primary_key=True)
    subject_title = models.CharField(\
            max_length=60, verbose_name="Subject Title")
    subject_short_name = models.CharField(\
            max_length=3, verbose_name="Abbriviation")
    subject_dep = models.ForeignKey('Department',\
            on_delete=models.CASCADE, verbose_name="Branch")
    subject_sem = models.SmallIntegerField(\
            choices=sem_choices, null=True)

    def __str__(self):
        return self.subject_title


class Teacher(models.Model):
    teacher_name = models.CharField(max_length=40)
    dept = models.ForeignKey('Department', \
            on_delete=models.CASCADE)
    subjects = models.ManyToManyField('Subject', \
            verbose_name="Subjects")

    def __str__(self):
        return self.teacher_name


class Class(models.Model):
    current_sem = models.PositiveSmallIntegerField(\
            choices=sem_choices)
    branch = models.ForeignKey('Department', \
            on_delete=models.CASCADE)
    batch = models.CharField(max_length=1)
    class_name = models.CharField(max_length=10, blank=True)
    timeTable = models.TextField()
    subjects = models.ManyToManyField('Subject', \
            verbose_name='Subjects', through='TaughtBy')

    def save(self, *args, **kwargs):
        self.class_name = str(self.branch) + " " + \
                str(self.current_sem) + str(self.batch)
        super(Class, self).save(*args, **kwargs)

    def get_subs(self):
        subs = [subject.subject_short_name for subject \
                in self.subjects.all()]
        return subs

    def get_tt(self):
        tt_string = str(self.timeTable)
        tt_string = tt_string.splitlines()
        days = ['monday', 'tuesday', 'wednesday',
                'thursday', 'friday']
        tt_dict = {}
        for i in range(5):
            tt_dict[days[i]] = tt_string[i]

        return tt_dict

    def __str__(self):
        return str(self.class_name)

    class Meta:
        verbose_name_plural = "Classes"
```

5.1.4 Base html template

A basic html template for a page

```
<div class='fullpage'>
    <div class='leftpane pane'>
        <div class='card small heading' id='lheading'>
            Navigation
        </div>
        <div class='navel' id='home'>
            <i class="fa fa-home" aria-hidden="true">
            </i><br>
            Home
        </div>
        <div class='navel' id='notes'>
            <i class="fa fa-sticky-note" aria-hidden="true">
            </i><br>
            Notes
        </div>
        <div class='navel' id='attendece'>
            <i class="fa fa-child" aria-hidden="true">
            </i><br>
            Attendence
        </div>
        <div class='navel' id='calender'>
            <i class="fa fa-calendar" aria-hidden="true">
            </i><br>
            Calender
        </div>
        <div class='navel' id='about'>
            <i class="fa fa-info-circle" aria-hidden="true">
            </i><br>
            About
        </div>
    </div>
    <div class='midpane pane'>
        <div id='mheading'>
            <i class="fa fa-bars" aria-hidden="true"
                id="hamburger-menu-button"></i>
            <img id="profile-image" src="" alt="profile">
            <div id='profile-name'></div>
            <i class="fa fa-asterisk"
                aria-hidden="true"
                id="upcoming-menu-button"></i>
        </div>
        <div id='mcontent'>
        </div>
    </div>
    <div class='rightpane pane'>
        <div class='card small heading' id='rheading'>
            Upcoming
        </div>
        <div class='content' id='rcontent'>
        </div>
    </div>
</div>
```

5.1.5 Sample module in js

Our js file in a modular design and the below show is the code to initialize a user account.

```
InitializeUser = function(user){
    this.class_name = 'initialize_user';
    this.user = user;
}
InitializeUser.prototype.get_data_from_server =
    function(callback){
    var self = this;
    if (callback === undefined) { callback=function(){} }
    $.get(server_address+'/events/'+fuid,
        function(upcoming_events){
        self.events = upcoming_events['data'];
        callback()
    })
}
InitializeUser.prototype.init = function(events){
    var self = this
    this.get_data_from_server(function(){
        self.populate_user_profile();
        self.populate_upcoming();
        self.click_handlers();
    })
}
```

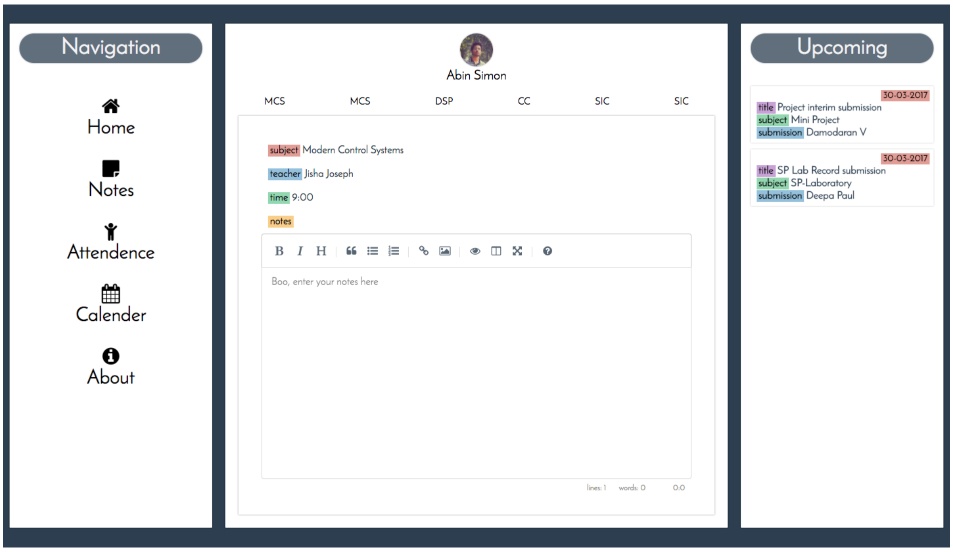
```
InitializeUser.prototype.populate_user_profile = function(){
    $('#profile-name').html(this.user['name']);
    $('#profile-image').attr('src', this.user['photo']);
    $('#popup-profile-name').html(this.user['name']);
    $('#popup-profile-image').attr('src', this.user['photo']);
    $('#popup-profile-email').html(this.user['email']);
}
InitializeUser.prototype.populate_upcoming = function(){
    var self = this;
    htmlstr = '';
    for( var i=0, len=this.events.length; i<len; i++ ){
        htmlstr +=
            "<div class='card small event' data-event="+i+">"+
                "<div class='card tiny date-card'>"+
                    this.events[i]['due']+
                "</div>"+
                "<div class='card tiny subject-card'>"+
                    "<span class='span violet name'>title"+
            "</span><span class='span white span-content'>"+
            this.events[i]['name']+ "</span>"+
                "</div>"+
                "<div class='card tiny subject-card'>"+
                    "<span class='span green name'>subject'"+
            "</span><span class='span white span-content'>"+
            this.events[i]['subject']+ "</span>"+
                "</div>"+
                "<div class='card tiny subject-card'>"+
            "<span class='span blue name'>submission"+
        "</span><span class='span white span-content'>"+
            this.events[i]['to']+ "</span>"+
                "</div>"+
            "</div>"
    }
    $('#rcontent').html(htmlstr);

    // Now set up the click handlers
    event_cards = $('.event');
    for(var i = 0, len = event_cards.length; i<len; i++){
        $(event_cards[i]).click(function(){
            event_data = self.events[$($(this)[0])
                .data('event')]
            el_main = $('#event-popup').children()
            el_date = $($(el_main[1]).children()[1])
                .text(event_data['due'])
            el_title = $($(el_main[2]).children()[1])
                .text(event_data['name'])
            el_subject = $($(el_main[3]).children()[1])
                .text(event_data['subject'])
            el_submission = $($(el_main[4]).children()[1])
                .text(event_data['to'])
            el_desc = $($(el_main[5]).children()[3])
                .text(event_data['description'])
            $($('#event-popup').parent())
                .css('display', 'flex');
        });
    }
    $($('#event-popup').parent()).click(function(){
        $($('#event-popup').parent())
            .css('display', 'none');
    });
}
InitializeUser.prototype.click_handlers = function(){
    // User logout
    $('#popup-profile-signout-button').click(function(){
        var auth2 = gapi.auth2.getAuthInstance();
        auth2.signOut().then(function () {
            console.log('User signed out.');
        });
        $('#login-popup').css('display', 'flex');
        $($('#signout-popup').parent())
            .css('display', 'none');
    });
    // Handle sinout popup remove click
    $($('#signout-popup').parent()).click(function(){
        $($('#signout-popup').parent())
            .css('display', 'none');
    });
    $('#signout-popup').click(function(e){
        e.stopPropagation();
    });
    // Open up the signout popup
    $('#profile-image').click(function(){
        $($('#signout-popup').parent())
            .css('display', 'flex');
    });


    //other handles (menu items)
    $('#home').click(function(){
        new Home().init()
    });
    $('#notes').click(function(){
        new NotesView().init()
    });
    $('#about').click(function(){
        new AboutView().init()
    });
    $('#attendece').click(function(){
        new AttendenceView().init()
    });
    $('#calender').click(function(){
        new CalenderView().init()
    });
}
```

5.1.6 Sample CSS file

In here we define the styling for the elements

```
.fullpage{
    width: 100%;
    height: 100%;
    overflow: hidden;
}
.pane{
    padding: 10px;
    margin: 10px;
    margin-top: 30px;
    margin-bottom: 30px;
    float:left;
    border-radius: 1px;
    box-shadow: 0px 0px 2px 0px rgba(0,0,0,0.75);
    background-color: #fff;
}
.leftpane{
    width: 20vw;
    height: calc(100% - 80px);
    transition-property: width, visibility;
    transition-duration: 0.5s;
    z-index: 100;
    overflow: auto;
}
.midpane{
    width: calc(60vw - 120px);
    height: calc(100% - 80px);
    transition-property: width;
    transition-duration: 0.5s;
    overflow: hidden;
}
.rightpane{
    width: 20vw;
    height: calc(100% - 80px);
    transition-property: width;
    transition-duration: 0.5s;
    z-index: 100;
    overflow: auto;
}
@media screen and (max-width: 1050px){
    /* Remove navigation */
    .leftpane{
        display: none;
    }
    .rightpane{
        width: 230px;
    }
    .midpane{
        /* width: calc(100vw - 40px); */
        width: calc(100vw - 320px);
    }
}
```

5.2 Screenshots



subject Modern Control Systems
teacher Jisha Joseph
time 9:00
notes

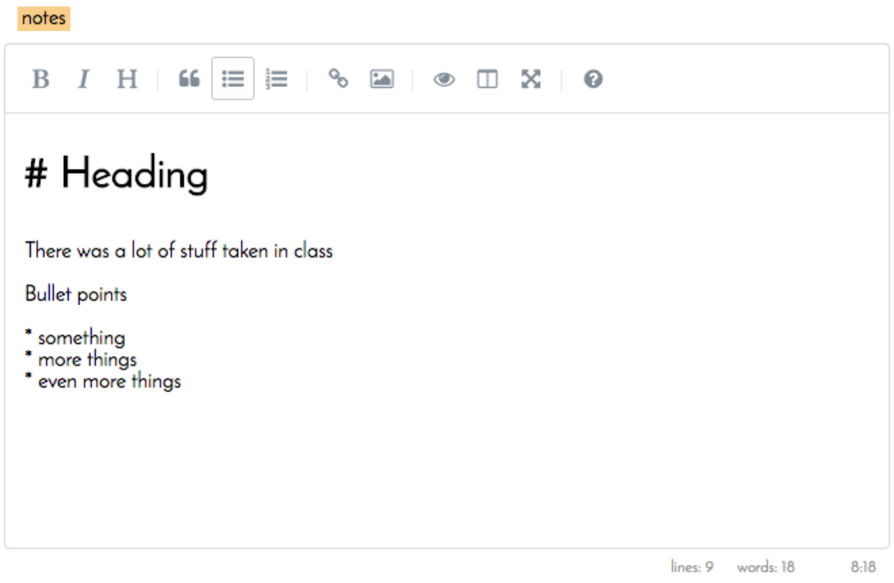
Boo, enter your notes here

lines: 1 words: 0 0:0

30-03-2017
title Project interim submission
subject Mini Project
submission Damodaran V

30-03-2017
title SP Lab Record submission
subject SP-Laboratory
submission Deepa Paul

Figure 5.1: Landing page



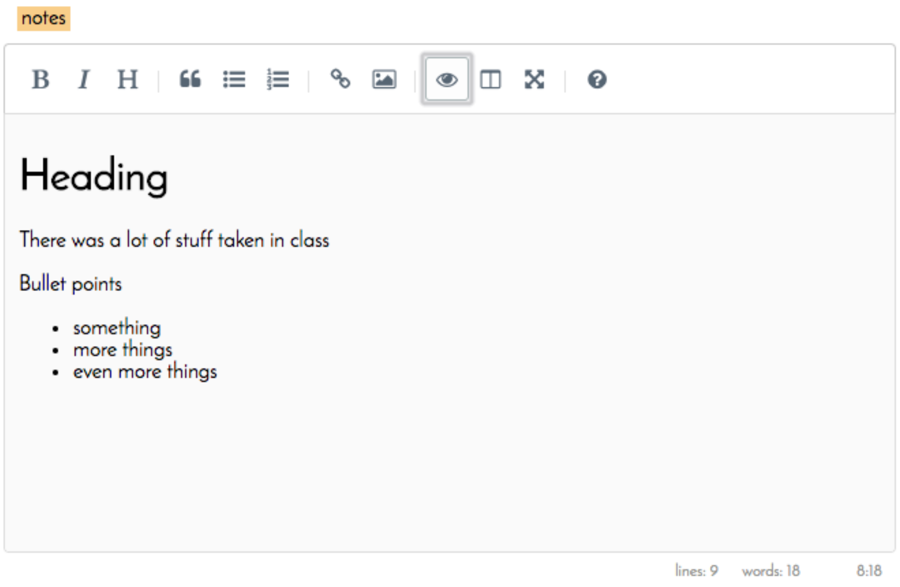
Heading

There was a lot of stuff taken in class

Bullet points

* something
* more things
* even more things

Figure 5.2: Write notes in markdown



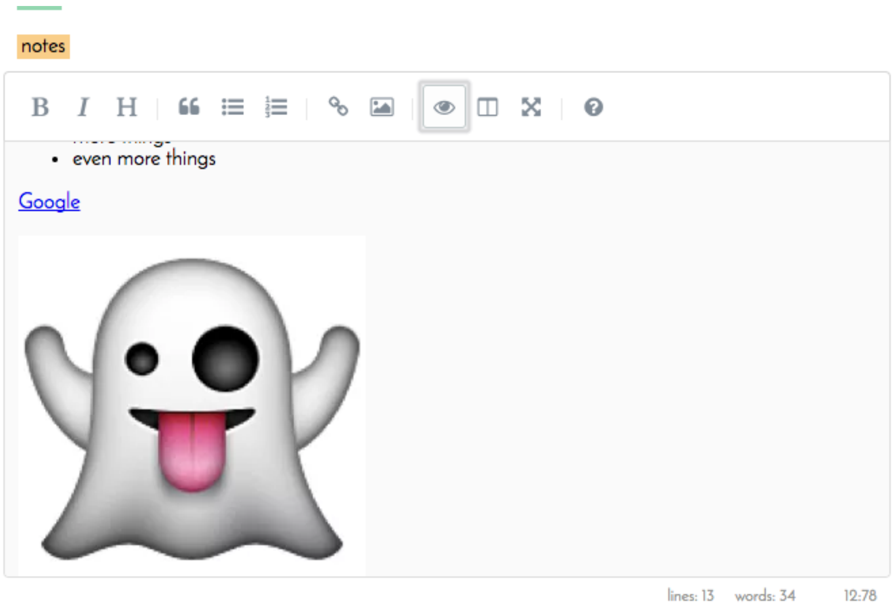
Heading

There was a lot of stuff taken in class

Bullet points

- something
- more things
- even more things

Figure 5.3: Compiled the written notes to actual contet in place

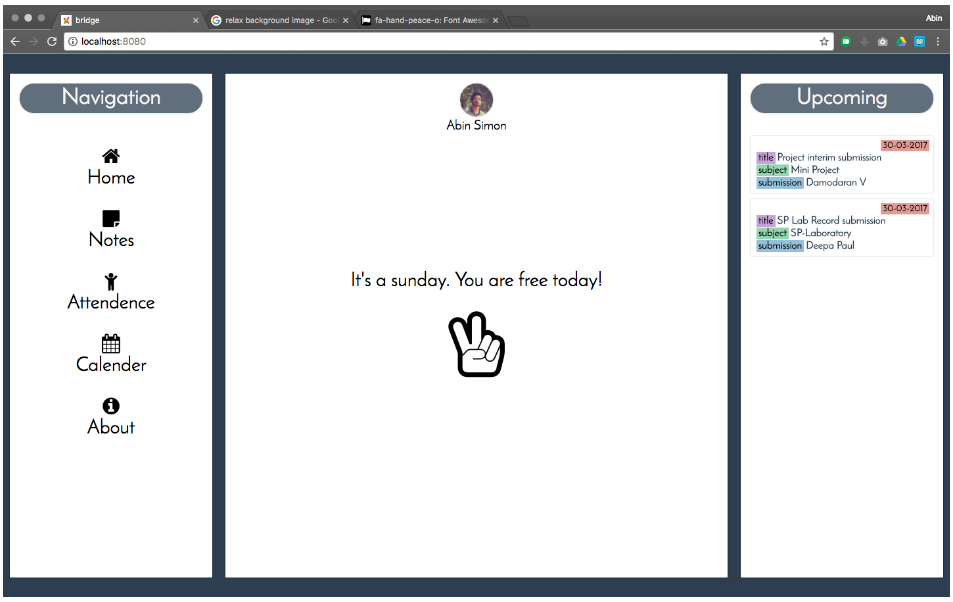


- even more things

Google

lines: 13 words: 34 12:78

Figure 5.4: You can even have notes with images and links



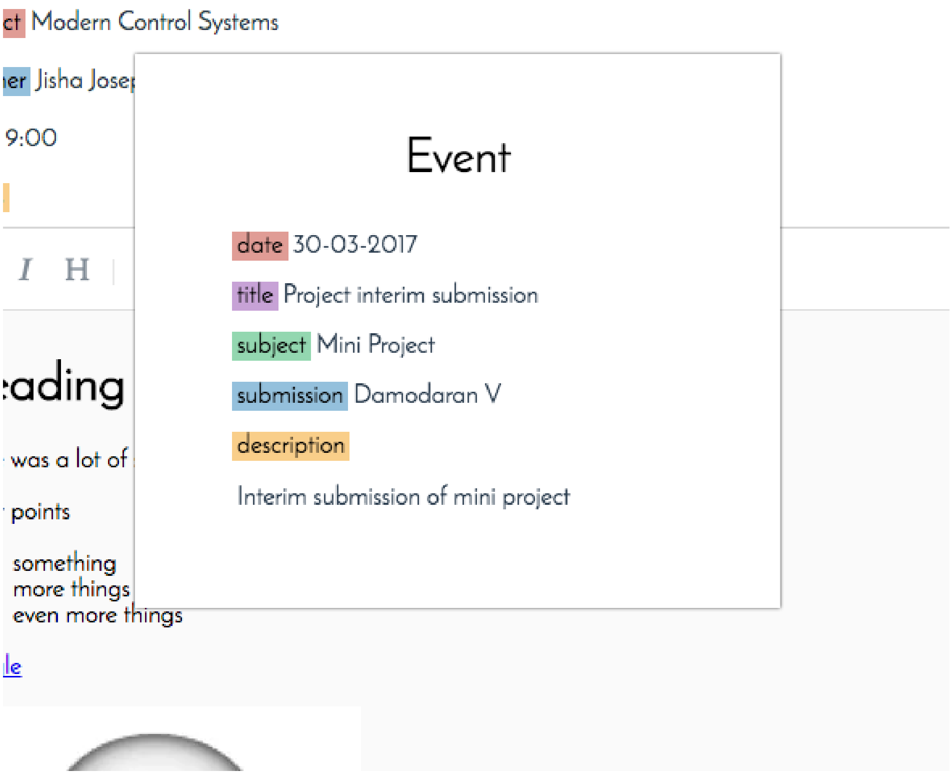
Abin Simon

It's a sunday. You are free today!

30-03-2017
title Project interim submission
subject Mini Project
submission Damodaran V

30-03-2017
title SP Lab Record submission
subject SP-Laboratory
submission Deepa Paul

Figure 5.5: Landing page on a holiday



ct Modern Control Systems

ier Jisha Josep

9:00

I H

:ading

was a lot of

points

something
more things
even more things

le

Event

date 30-03-2017

title Project interim submission

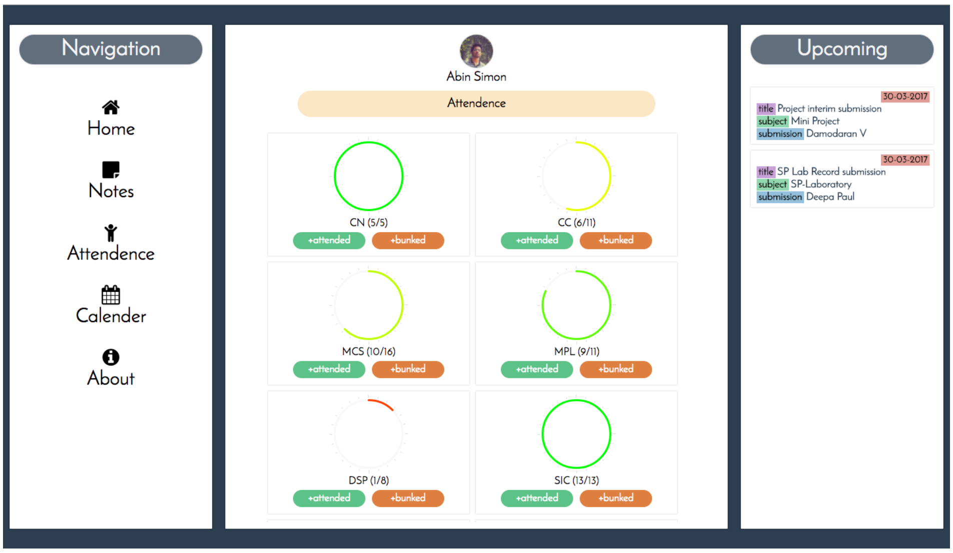
subject Mini Project

submission Damodaran V

description

Interim submission of mini project

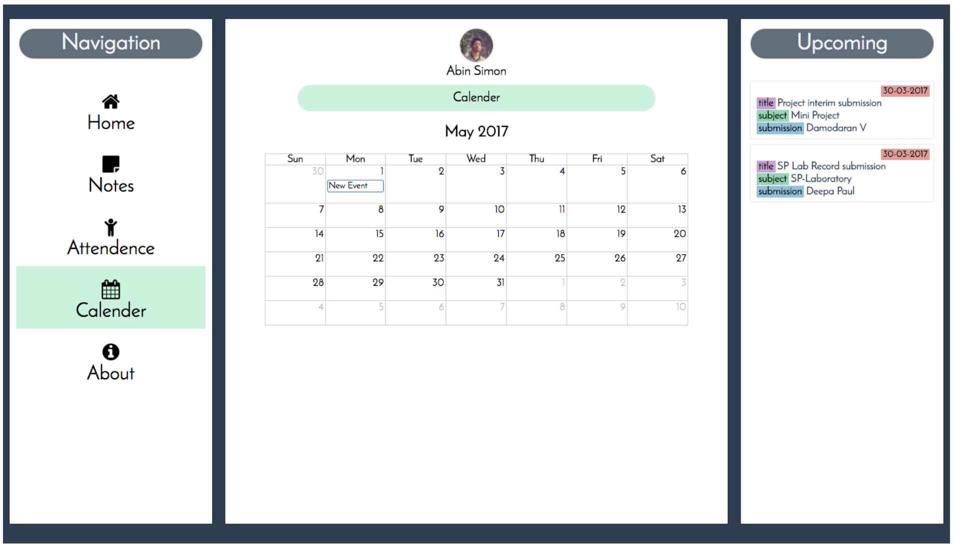
Figure 5.6: Detailed information about upcoming events



30-03-2017
title Project interim submission
subject Mini Project
submission Damodaran V

30-03-2017
title SP Lab Record submission
subject SP-Laboratory
submission Deepa Paul

Figure 5.7: View and update attendace



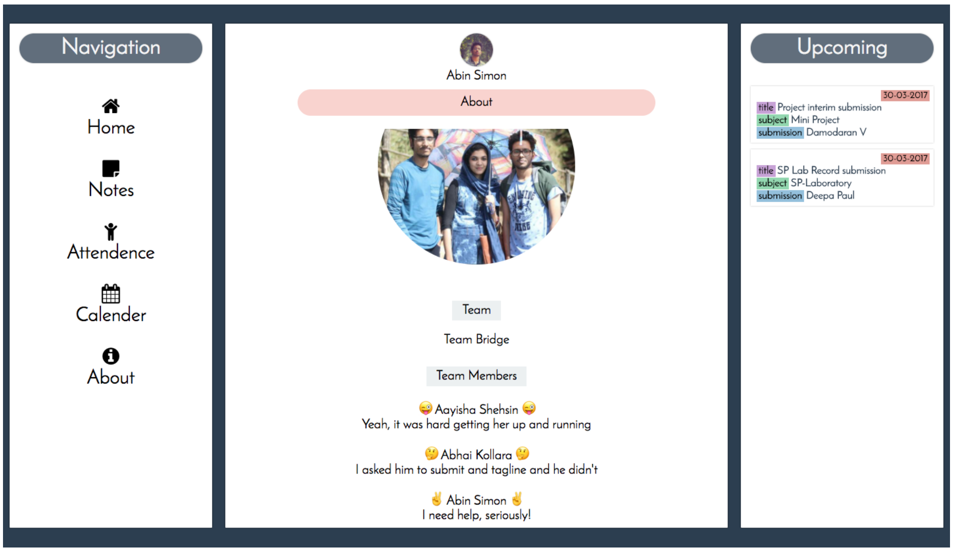
May 2017

Sun	Mon	Tue	Wed	Thu	Fri	Sat
30	1 New Event	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31	1	2	3
4	5	6	7	8	9	10

30-03-2017
title Project interim submission
subject Mini Project
submission Damodaran V

30-03-2017
title SP Lab Record submission
subject SP-Laboratory
submission Deepa Paul

Figure 5.8: Calendar entries for events



Abin Simon

About

Team

Team Bridge

Team Members

😜 Aayisha Shehsin 😜
Yeah, it was hard getting her up and running

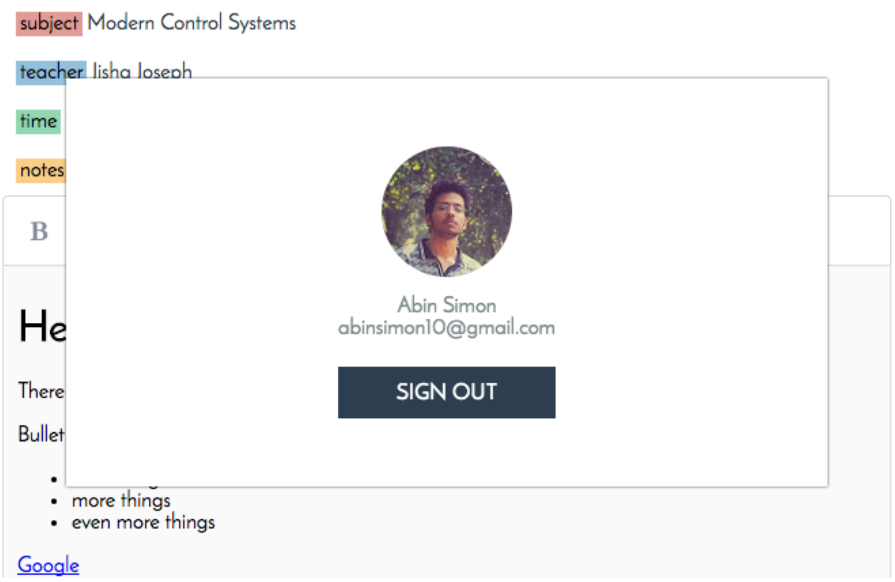
🤔 Abhai Kollara 🤔
I asked him to submit and tagline and he didn't

✌ Abin Simon ✌
I need help, seriously!

30-03-2017
title Project interim submission
subject Mini Project
submission Damodaran V

30-03-2017
title SP Lab Record submission
subject SP-Laboratory
submission Deepa Paul

Figure 5.9: About page



subject Modern Control Systems

teacher Jisha Joseph

time

notes

He

There

Bullet

-
- more things
- even more things

Google

Abin Simon
abinsimon10@gmail.com

Figure 5.10: Popup to sign out of the system

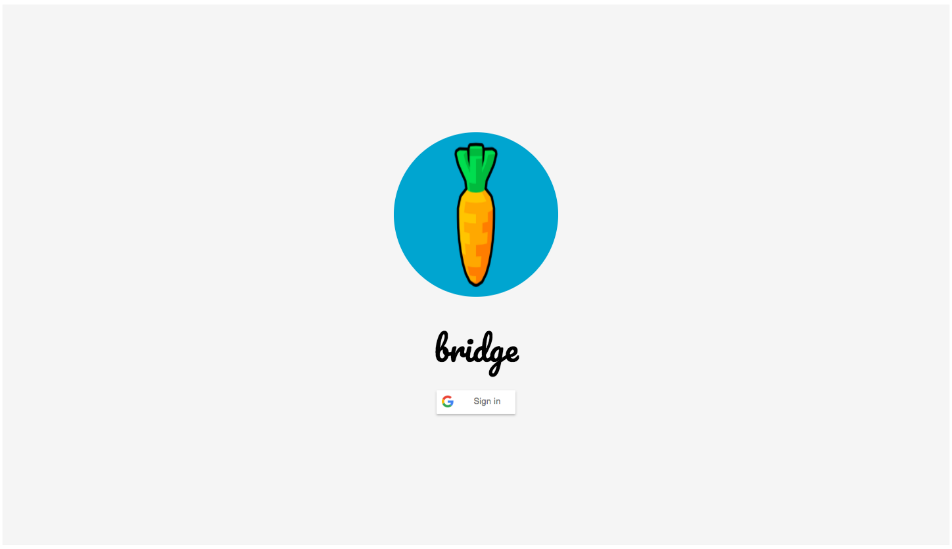
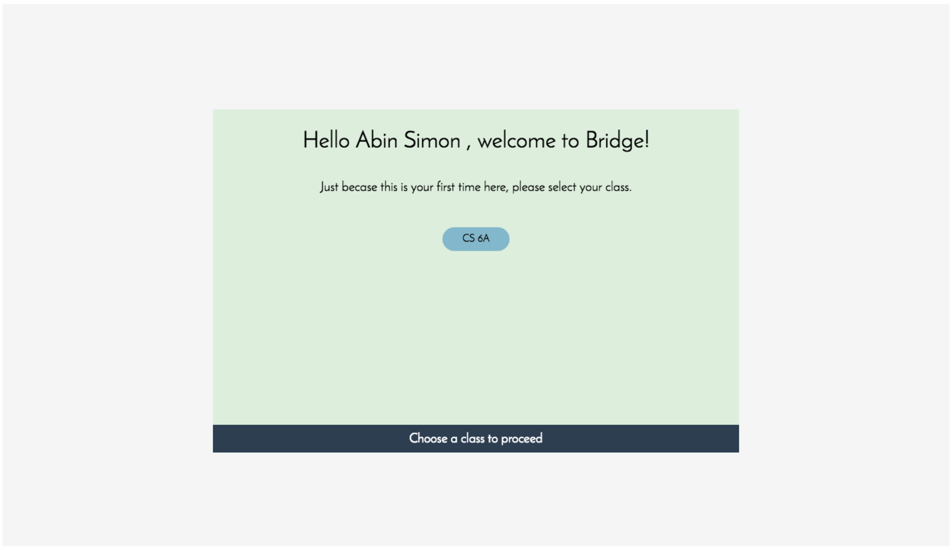
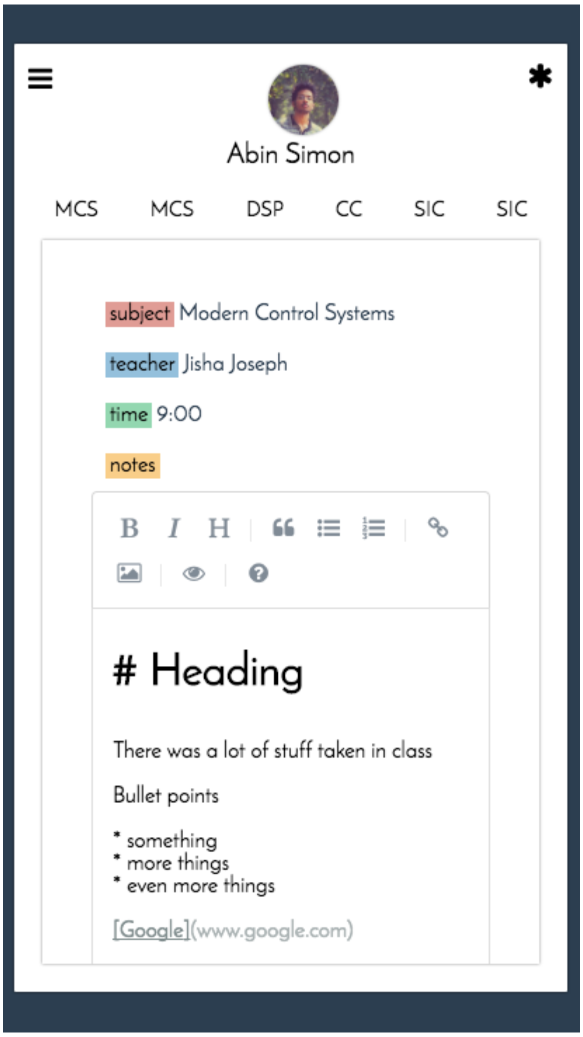


Figure 5.11: Interface for loggin into the app


Hello Abin Simon , welcome to Bridge!

Just becase this is your first time here, please select your class.

Figure 5.12: Interface if the user is a new one so that he can choose a class


Abin Simon

MCS MCS DSP CC SIC SIC

subject Modern Control Systems

teacher Jisha Joseph

time 9:00

notes

```
# Heading

There was a lot of stuff taken in class

Bullet points

* something
* more things
* even more things

[Google](www.google.com)
```

Figure 5.13: Landing page on mobile

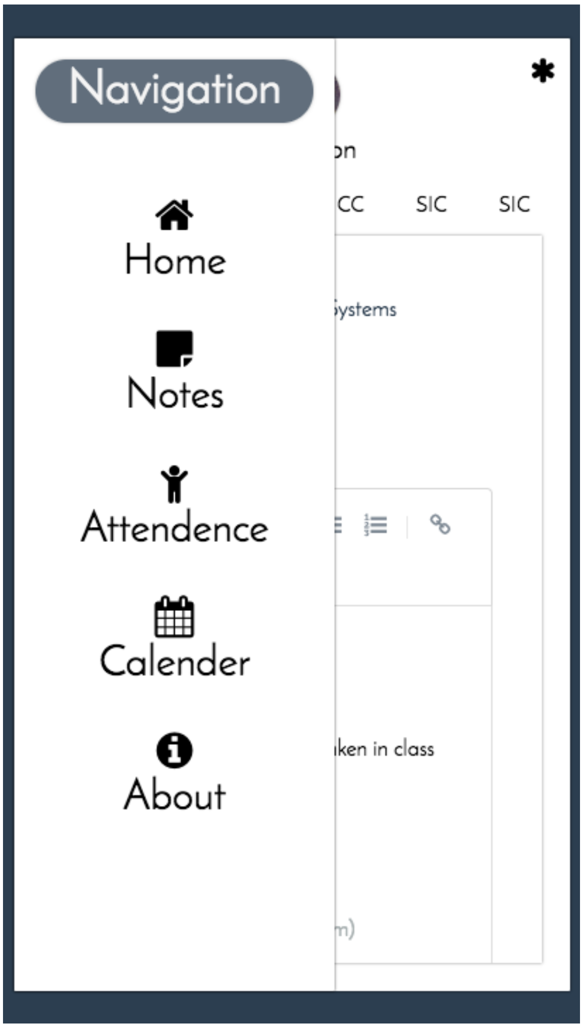
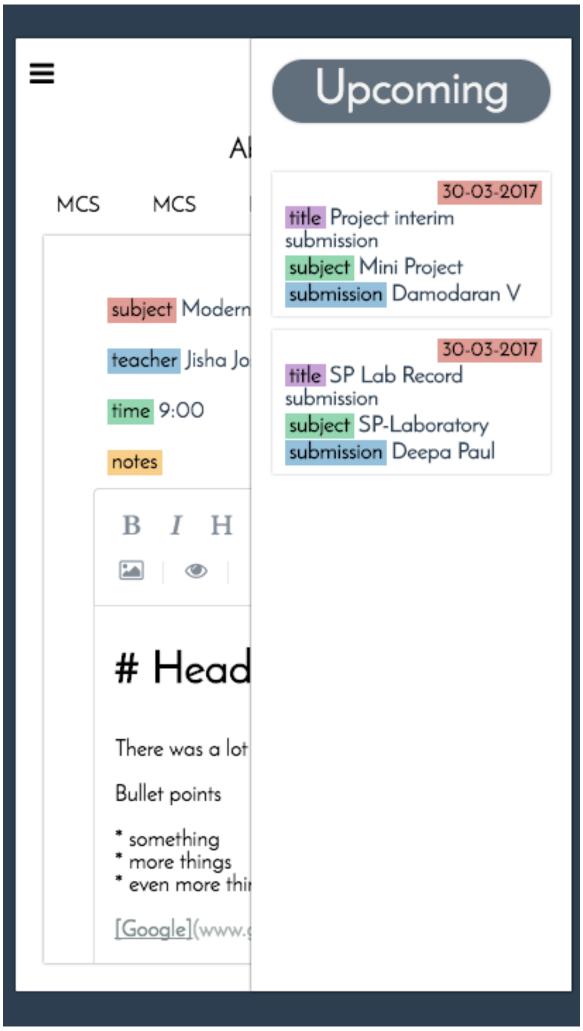


Figure 5.14: Navigation pane on mobile



Upcoming

30-03-2017
title Project interim submission
subject Mini Project
submission Damodaran V

30-03-2017
title SP Lab Record submission
subject SP-Laboratory
submission Deepa Paul

Al

MCS MCS

subject Modern
teacher Jisha Jo
time 9:00
notes

B I H

Head

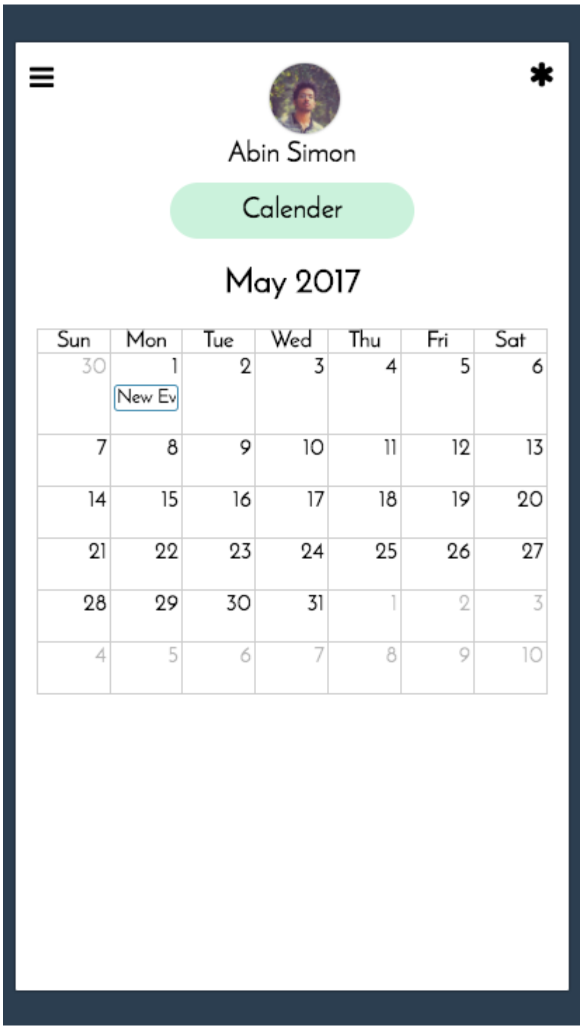
There was a lot

Bullet points

* something
* more things
* even more thi

[Google](www.

Figure 5.15: Upcoming event display on mobile



Abin Simon

Calender

May 2017

Sun	Mon	Tue	Wed	Thu	Fri	Sat
30	1 New Ev	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31	1	2	3
4	5	6	7	8	9	10

Figure 5.16: Calendar display on mobile

Chapter 6

System Testing

The aim of the system testing process was to determine all defects in our project. The program was subjected to a set of test inputs and various observations were made and based on these observations it will be decided whether the program behaves as expected or not.

Our Project went through four levels of testing

- Unit testing
- Integration testing
- System testing
- Component Interface Testing

6.1 Unit testing

Unit testing is undertaken when a module has been created and successfully reviewed. In order to test a single module we need to provide a complete en- vironment i.e. besides the module we would require. The procedures which belong to other modules that the module under test calls non local data structures that module access a procedure to call the functions of the mod- ule under test with appropriate parameters. Unit testing was done on each and every module that is described under module wise description.

6.2 Integration testing

In this type of testing we test various integration of the project module by providing the input. The primary objective is to test the module interfaces in order to ensure that no errors are occurring when one module invokes the other module.

6.3 System testing

System testing, or end-to-end testing, tests a completely integrated system to verify that it meets its requirements. For example, a system test might involve testing a logon interface, then creating and editing an entry, plus sending or printing results, followed by summary processing or deletion (or archiving) of entries, then logoff. In addition, the software testing should ensure that the program, as well as working as expected, does not also destroy or partially corrupt its operating environment or cause other processes within that environment to become inoperative (this includes not corrupting shared memory, not consuming or locking up excessive resources and leaving any parallel processes unharmed by its presence).

6.4 Component interface testing

The practice of component interface testing can be used to check the handling of data passed between various units, or subsystem components, beyond full integration testing between those units. The data being passed can be con- sidered as ”message packets” and the range or data types can be checked, for data generated from one unit, and tested for validity before being passed into another unit. One option for interface testing is to keep a separate log file of data items being passed, often with a timestamp logged to allow analysis of thousands of cases of data passed between units for days or weeks. Tests can include checking the handling of some extreme data values while other interface variables are passed as normal values. Unusual data values in an interface can help explain unexpected performance in the next unit. Com- ponent interface testing is a variation of black box testing, with the focus on the data values beyond just the related actions of a subsystem component.

Chapter 7

Future Scope

The project serves all its present requirements. Like any other system, there is always room for improvement. The system in the future expects to be more in every way. Automate more components and also create new components for both teachers and students alike.

Chapter 8

Conclusion

A conscious effort has been made while creating and developing the software package ,making use of existing tools, techniques and resource that would cater the demands .The project has shown us the finer aspect of database management system. While making the system, an eye has been kept on making it as user-friendly, as cost-effective as possible. As such one may hope that the system will be acceptable to any user and will adequately meet his/her needs. The entire project is menu assisted and highly interactive. On trial run the performance was found to be satisfactory. But as in case of any system development processes , there were are a number of shortcomings, there have been some shortcomings in the development of this system also. Thus project is still under modification.

References

[1] Learn python the hard way, `https://learnpythonthehardway.org/`

[2] Django official docs, `https://docs.djangoproject.com/en/1.11/`

[3] Django Girls Tutorial, `https://tutorial.djangogirls.org/en/index.html`

[4] W3 Schools, `https://www.w3schools.com/`

[5] Tutorials point, `https://www.tutorialspoint.com/`

[6] Stack Overflow, `http://stackoverflow.com/`

[7] Udacity, `https://in.udacity.com/`

[8] Coursera, `https://www.coursera.org/`