Analysis of Algorithms

Spring 2020

Members Details

Group ID	CS311S20PID32
Registration Number of Group Members	2018-CS-31 2018-CS-32
Section	A

Project Details

Project	
Project Title	Activity Scheduling Tool
Executive Summary	The purpose of the project is to find an optimal solution to the time tabling problem which is one of the highly constrained N-P hard problems. The need of this sort of a time tabling software arised as manually designing a time table takes too much time and effort and if overlap occurs among the timetable, the timetable is redesigned using hit and error methods which has very high time cost. So, in this project we are trying to develop such a software which will automatically and will decrease the effort and time required for the generation of a time table in accordance to the given input. The expected main input is about teachers, classes and subject's data along with the maximum workload of a teacher in a week to generate a valid time-table. The main constraints that this software should satisfy are that a teacher should not have a lecture in more than one class at the same time slot and a class should not have more than one lecture in a given time slot. The solution which we will get from this project should have to satisfy the above-mentioned constraints. The platform we used to develop this software is web application. The programming language used to implement this software is JavaScript. In the development of the UI, React.js is used while backend is implemented using node.js. To make it a multi user app login facility is provided and to store the data corresponding to the user, mongodb database is used locally. Project is currently run on local severs but we have planned it to deploy it using Heroku and we will also change the database from local to mongodb atlas. This projected is completed in 4 steps.

	• Step1 We have designed our algorithm for this problem by analyzing the problem deeply and designed the UI for the project. • Step2 In the second step our project we worked to improve the efficiency of our algorithm and calculated the time complexity of it. We also proved the correctness of our algorithm. • Step3 In this phase we implemented the algorithm on the backend and get the desired results by providing our dummy inputs. • Step4 Backend was implemented and Rest API's were developed to interact with database and frontend • Step5 Frontend was developed and iteration of backend with our UI is done in this step.
Business Case	
Outline the business need for the project	Designing of timetables is one of the most complicated problem. Designing a time table for a school or department in most of the institutions are still done by hand which has many drawbacks. Manual designing of time table takes too much time and is a difficult task. If the time table is to be generated for a bigger faculty, the optimal solution may contain clashes and to resolve these clashes a teacher may need to compromise on his schedule. So there arises a requirement of a software which should automatically generate the time table without any clashes.
End user of the product	In this project our intended audience is all the educational institutions. These institutions are schools, colleges and universities. Our main target in this project is the administration of our university.
Motivation for Project	As our faculty faces a stern task of designing the time table for each section in every new semester, we decided to provide them with an application which will generate an automated timetable which will ease off the burden of designing a time table by hand. Moreover, by choosing this project we were set to have exposure of how the

	projects are managed in industry as milestones and workflows were purely designed in accordance to the requirements of industry.
Description of the project objective(s)	One of the main objectives of this project is to develop an algorithm of lowest running cost which will generate a time table automatically and to integrate that algorithm with some sort of graphical user interface. The other main and important objective is to develop a full stack web application where a database should be used to make it a multi user application. Login and sign up facility should be provided for users to make a secure and permanent storage system. User authentication is also implemented. In this application our objective is to store all the records entered by the user and to show them on the UI.
State the level of impact expected should the project proceed and implications of not proceeding	This project is open sourced which means that there is no cost whatsoever in using this project assuming that user have access to internet and local machine to run this software. Use of such projects in educational institutions should be encouraged considering that its economical, efficient and the biggest advantages of reduced workload of faulty in designing the software manually.
Functional Requirements	The main requirement of this project is to generate a timetable which should be clash free. The functional requirements of this project are further divided into user and admin requirements. 1. Admin Requirements • The software should handle new users i.e. save the credentials of new user successfully. • Input should be validated. • Whenever a new record is inserted, it should be shown in the view records. • Login should be authenticated. User with false information should not be allowed to log into the system. • Responsive UI for all the platforms i.e. laptop, mobile. • Show error on UI if a user violates any of validation • Responsive buttons. • Get the previously saved data of a user. • Display the generated time table in tables on UI. 2. User Requirements • User provide email and his information to register into the system.

	• User should provide details about subjects, teachers and classes.
Benefits	
What benefits are expected/ anticipated?	• Academic Benefits After successful implementation of this project, we are able to have a fair share of grip in developing algorithms for the real-world problems. It was excellent experience of working in phases and meeting all the deadlines and this brought a great exposure of managing such complex projects. As algorithm analysis is a key subject in the field of computer science, we are able to understand it far better after doing this project. Moreover, we have a great experience and learning as we have developed a full stack application first time. • Industrial Benefits It will reduce the errors that are expected in the manual time table. It will reduce the efforts of the human hands that is required to handle these errors and to use each slot. It assures the optimal solution which satisfies all the given constraints. As this application is open sourced so it decreases the costs at industrial level as some institutions tend to use all online time table generating software which are paid.
Implementation Details	
Link to GitHub Repository	https://github.com/ghulamghousdev/CS311S20PID32
Total Number of commits in repository before 5th August 2020	229
Exact contribution of each member	There is no such distribution of work among the group members. Both group members mutually completed each milestone by mutually working on it. Both the group members lead the way in completing the milestones according to the respective skill sets i.e. in analysis and correctness of algorithm, pseudo code writing and report writing 2018-CS-31 lead the way and put bit more effort then the other group member. Similarly, 2018-CS-32, lead the way while developing the UI as he has superior skill set in that respective field and solely managed the state integration in the UI and handling of http requests. Designing of the REST API's and Models for the database was designed by both of the group members. 2018-CS-31 lead the way

	while implementing the algorithm on the backend.

Commits in GitHub repository by each member

Member Registration No.	Total Commits
2018-CS-31	110
2018-CS-32	119

Details of commits

Sr. No.	Details of commit	Date	Member Reg No.
1	Implemented Rest API for User Management. Using this API, a user will be able to save his credentials in database.	3-July-2020	2018-CS-32
2	Defined Schema and Model for User	3-July-2020	2018-CS-32
3	Setup Express server	3-July-2020	2018-CS-32
4	README file is written	4-July-2020	2018-CS-31
4.1	Pseudo Code	4-July-2020	2018-CS-31
5	Time Complexity Analysis	8-July-2020	2018-CS-31
6	Correctness of Algorithm	8-July-2020	2018-CS-31
7	Algorithm implementation	14-July-2020	2018-CS-31
8	Defined Schema for Subject, Class and Teacher	17-July-2020	2018-CS-31
9	Added Routes for Subject, Class and Teacher Model	18-July-2020	2018-CS-32
10	React project is initialized	26-July-2020	2018-CS-32
11	Added login, signup components	27-July-2020	2018-CS-32
12	Added Dashboard Component	29-July-2020	2018-CS-32
13	Added Components for Add Subject, Add Teacher for adding records on front end and Class list, Class List Item component to view the class records on UI	29-July-2020	2018-CS-32
14	Added Add Slots Component in front end and Defined Schema and Model on Backend	30-July-2020	2018-CS-31
15	Added components to view records for Subjects and Teachers records	31-July-2020	2018-CS-31
16	Added Component to view slots and	02-Aug-2020	2018-CS-31

	REST API to get data from database for algorithm execution		
17	Added Guidelines for Project Configuration in README	04-Aug-2020	2018-CS-31
18	Handled HTTP requests	04-Aug-2020	2018-CS-32
19	Added Page for output	12-Aug-2020	2018-CS-32
20	Resolved Issue of Routing	14-Aug-2020	2018-CS-32
21	User manual to resolve issue created by the testing team	15-Aug-2020	2018-CS-31

Have you used build in algorithms or you have implemented yourself?	We took help from multiple forums and used their inputs on this problem. We found a source where some part of algorithm is provided, we took it and further enhanced the functionality of that algorithm. First it was used to generate the time table only for one section but after modification it is now able to generate multiple time table for multiple sections.
Formats of input	Input attributes which are required for this project are subjects, teachers, class names and slots. Moreover, to use this project, a user credentials are also required to sign up into the system. The input format for user input is described in the below paragraph. We are taking input separately for each attribute i.e. for subjects, classes, teachers and slots.

Page	Input Fields and type
Sign up	First Name = text, Second Name = text, Email = text, Password = text
Login	Email = text, Password = text
Subjects	Subject ID = text, Subject Name = text, Credit Hours = number, Contact ours = number, Labs = number
Classes	Class Name =text, Session = text, Section = text
Teachers	First Name =text, Second Name = text, Reg Number = text, working hours = number
Slots	Teacher Name = text, Subject Name = text,

<table>
<tr><td></td><td><table><tr><td></td><td>Section = text,
Session = text,
Lectures = number</td></tr></table></td></tr>
<tr><td>Validations</td><td>Following validations are used on input.
Subject Name, Teacher Name should be strings.
Validation on format of email is also implemented.
Password should have min length 5 and max 15.
Credit should not be greater than 3 and not less than 0
Contact Hours should not be greater than 3 and less than 0.
Subject Code and Reg Number of teachers should be unique.
Working Hours should not be less than 0.
Credit Hours, contact hours, working hours and labs should be numeric values.
Password and Confirm password should be same.</td></tr>
<tr><td>Format of output</td><td>After the execution of algorithm on the provided input, our algorithm will return an array that will further contain the nested arrays for each class. In each nested array for class, there will be arrays equal to number of working days. We will show the time table on UI in the form of tables. Each table will represent time table of one class. Each row will represent a working day and each slot will represent a time slot. In each cell we will show the assigned teacher name and assigned subject name.</td></tr>
<tr><td>Deployment</td><td>We have not deployed the project yet but we plan to deploy it on Heroku.</td></tr>
<tr><td colspan="2">Details of algorithms</td></tr>
<tr><td colspan="2">Algorithm Description

As Time table generating problem is one of the N-P hard problems, so it is difficult to get a optimal solution. The algorithmic approach which we used in this project is Heuristic Approach. All the hard constraints are dealt with by using Constraint-based Programming. The algorithm has two functions, one main function generateTimeTable and a supportive function randDay which is used in the main function. Features and Constraints implemented can be found here.
Our algorithm takes multiple inputs which are listed below:

instances:
Data Structure which stores info about provided slots to be organized i.e.
[[Ti, Ci, Si, LTi, Li],.....,[Tn, Cn, Sn, LTn, Ln]]

Here</td></tr>
</table>

T = Teacher
C = Class
S = Subject
LT = noOfLectures
L = Labs
ins = which keeps record of lecture assigned & will be added in the generate function according to number of slots given

givenSlots:
Data structure which stores info about GivenSlots on each day i.e.
[3,4,5,3,2]

classes:
Data structure which stores info about Classes(classes) i.e.
["A", "B"]

teachers:
Data structure which stores info about Teachers i.e.
["T1", "T2"]

Our algorithms generate time table section wise mean it handles time table of section at a time and check in the provided slots. If it founds a slot related to the class, it adds that instance to the section instances data structure. This was done in first half. In second half, again a section is selected and iterating through the slots on the given day and the section instance, a slot in time table is assigned. And the end of the algorithm we get a array which have nested arrays. Each nested array represents time table of a section.

- **Pseudo Code**
-

Input:
In this algorithm we will be giving following input to get desired results.

instances:
Data Structure which stores info about provided slots to be organized i.e.
[[Ti, Ci, Si, LTi, Li],.....,[Tn, Cn, Sn, LTn, Ln]]

Here
T = Teacher
C = Class
S = Subject
LT = noOfLectures
L = Labs
ins = which keeps record of lecture assigned & will be added in the generate function according to number of slots given
givenSlots:

Data structure which stores info about GivenSlots on each day i.e
[3,4,5,3,2]

classes:

Data structure which stores info about Classes(classes) i.e
["A", "B"]

teachers:

Data structure which stores info about Teachers i.e
["T1", "T2"]

Variables used in the algorithm

- sectionInstances: data structure to store info about each section
- TT: data structure which is initialized with all given slots with 0 and further on the variable containing info about lecture replaces zero which is decided and given that specific slot
- teacherTT: it stores info about each teacher and the slot which he is assigned a lecture
- numOfDays: it stores total working days;
  Flags & Counters to keep track of clashes
- regenerateTimeTableCountSec
- regenerateTimeTableFlagSec: flag to check if there comes any clash
- regenerateTimeTableListSec: Keeps record of input which causes clash
- timeTableNotPossibleCount: keeps count of how many time time table generation fails on specific input
- impossible: it says that it is impossible to generate time table with given data

Generate-Time-Table(instances, givenSlots, classes, teachers){

```
let i, j, k, numOfDays = 0
for (i = 1 to givenSlots.length){
   if givenSlots[i] > 0
   add 1 to numOfDays
}

initialize teachers, classes data structure upto number of given slots

for i in classes
      for j in instances
            for k in instances[j].classes
                  if(instances[j].classes[k] == classes[i])
                        instances[j][ins] = []
                        add instances[j] to secInstances at classes[i]
regenerateCountSec = 0
regenerateFlagSec =false
regenerateListSec = []
```

```
notPossibleCount = 0
impossible = false

for i in classes
        if impossible flag is true then return "Table not possible"  & break
         notPossible = false
         currentTT = [][]
         regenerateCountSI = 0
         regenerateFlagSI = false
         regenerateListSI = []

        for j in secInstances[classes[i]]
                 availableSlots = []

                 for day in givenSlots
                          let declare an empty data structure daySlots
                          for slot in givenSlots[day]
                                   if regenerateFlagSI is true then make slot flag true
                                            for a in regenerateListSI.slot
                                                     make dumFlag false
                                                     for b in slot
                                                              if slot at b is equal to regenerateListSI.slot at a
then make dumFlag true & break
                    if dum flag is false then make slot flag false and break

                 if( ( (!slotFlag) || (day != regenerateListSI.day)) &&
(teacherTT[secInstances[classes[i]][j].teacher][day][slot] == 0) && (currentTT[day][slot] == 0))
                    then add slot to daySlots;
                 regenerateFlagSI = false
              else if regenerateFlagSec is true then make slot flag true
                 for a in regenerateListSI.slot
                    dumFlag = false
                    for b in slot
                          if(slot[b] == regenerateListSI.slot[a]) then make dumpFlag True & break
                    if dumFlag is false then make slotFlag = true & break
                                             if(( (!slotFlag) || (day != regenerateListSec.day))
&&(teacherTT[secInstances[classes[i]][j].teacher][day][slot] == 0) &&  (currentTT[day][slot] == 0))
                    then make regenerateFlagSec  false
                                        elseif((teacherTT[secInstances[classes[i]][j].teacher][day][slot] == 0)
&& (currentTT[day][slot] == 0))
                 add slot to daySlots and then add daySlots to availableSlots;
        eachDay = secInstances[classes[i]][j].numLectures / numDays
        extraDays = secInstances[classes[i]][j].numLectures % numDays
        for i in range(numDays)
                          if there exits an extra day then add eachDay and plus 1 day to count and
decrement extraDays by 1 & repeat else make count equal to each day
           flag = true , radCount = 0
                          while(flag)
                                   const buffer = ranD(availableSlots, count)
                             if((buffer != undefined) && (buffer != null) && (buffer.day != undefined)
```

```
&& (buffer.slot!= undefined) &&  (buffer.day >= 0) && (buffer.day < givenSlots.length) &&
(buffer.slot.length == count))
                secInstances[classes[i]][j].push(buffer.day,ret.slot)
                for z in buffer.slot
                currentTT[buffer.day][buffer.slot[z]] = secInstances[classes[i]][j]
                teacherTT[secInstances[classes[i]][j].teacher][buffer.day][buffer.slot[z]] =
secInstances[classes[i]][j]
                                        availableSlots[buffer.day] = []
                flag = false
            else if radCount is less then 10 increment is by 1
            else if regenerateCountSI is greater then 100 then
                make regenerateSI and regenerateFlagSI to  true & flag to false
                        regenerateCountSI = regenerateCountSI + 1
                        regenerateListSI = secInstances[classes[i]][j].mapp[0]
                                        for y in secInstances[classes[i]][j].mapp
                                for w in secInstances[classes[i]][j].mapp.slot

currentTT[secInstances[classes[i]][j].mapp[y].day][secInsances[classes[i]][j].mapp[y].slot[w]] = 0

teacherTT[secInstances[classes[i]][j].teacher][secInstances[classes[i]][j].mapp[y].day][secInstances[clas
ses[i]][j].mapp[y].slot[w]] = 0
                                        secInstances[classes[i]][j].mapp = []
                j--
            else if regenerateCountSec is less then 100)
                make regenerateSec & regenerateFlagSec to true and flag to false and also make
regenerateCountSI = 0 and inc regenerateCountSec by 1
                                regenerateListSec = secInstances[classes[i]][0].mapp[0]
                                        for x in secInstances[classes[i]]
                    for y in secInstances[classes[i]][x].mapp
                                                for w in secInstances[classes[i]][x].mapp.slot

teacherTT[secInstances[classes[i]][x].teacher][secInstances[classes[i]][x].mapp[y].day][secInstances[cla
sses[i]][x].mapp[y].slot[w]] = 0

                                for x in secInstances[classes[i]]
                                    secInstances[classes[i]][x].mapp = []
                                i--
            else
                if notPossibleCount is less 1000 then
                                        make flag to false and notPossible to true also make
regenerateCountSec = 0 and inc notPossibleCount by 1
                                                let i= -1, TT = [], teacherTT = {}, secTT = {},
currentTT = []
                    for u in classes
                                                for v in instances
                                                    for w in instances[v].classes
                                                        if(instances[v].classes[w] == classes[u])
                                                                instances[v]["mapp"] = []
                        secInstances[classes[u]].push(instances[v])
                                                            else make impossible to true and flag to false
```

```
            if(impossible ||  notPossible || regenerateFlagSec || regenerateFlagSI) then break
          if( impossible ||  notPossible || regenerateFlagSec) then break
           if( (!impossible) && (!regenerateFlagSec) && (!notPossible))
          add currentTT to TT
          secTT[classes[i]] = currentTT
           if notPossible is true then make it false
           if impossible is true then return("Could not generate in this case, please refresh/restart")
      return TT
    }


ranD (slots, count) {
      let i, viableDays, slot
      for i in slots
            if slots[i].length >= count
                  add i in viable days
      if viableDays is empty then return null
      let buff = crypto.randomBytes(2);
      let n = parseInt(buff.toString('hex'),16)
      let index = n % (viableDays.length);
      let day = viableDays[index]
      for i = 0 to count {
            buff = crypto.randomBytes(2);
            n = parseInt(buff.toString('hex'),16)
            let s = n % slots[day].length;
            slot.push(slots[day][s]);
            slots[day].splice(s,1);
            return day, slot
      }
    }
```

- **Time Complexity Analysis of Pseudo Code**

We will analyze line by line and at the end will get sum of all the costs.

```
Generate-Time-Table(instances, givenSlots, classes, teachers){
  //We will use these variables to make our analysis easier to understand
  givenSlots = m
  noOfDays = d
  classes = c
  noOfnstances = n


  let i, j, k
  let numOfDays = 0 ---------------------->  1
  for (i = 1 to givenSlots.length){----------------------> d+1
    if givenSlots[i] > 0----------------------> d
```

```
    add 1 to numOfDays ----------------------> d
}

initialize teachers, classes data structure upto number of given slots

for i in classes   ----------------------->  c+1
      for j in instances   ----------------------->  nc + c
            for k in instances[j].classes----------------------> (n)(c)+nc
                  if(instances[j].classes[k] == classes[i]) ----------------------> nc
                        instances[j][ins] = []  ----------------------->nc
                        add instances[j] to secInstances at classes[i]  ----------------------> nc
regenerateCountSec = 0    ----------------------->  1
regenerateFlagSec =false   ----------------------> 1
regenerateListSec = []   ----------------------> 1
notPossibleCount = 0    ----------------------->   1
impossible = false      ----------------------->  1

for i in classes   ----------------------->c+1
      if impossible flag is true then return "Table not possible"  & break  ----------------------> c+c+c
       notPossible = false    ----------------------> c
       currentTT = [][]    ----------------------->   c
       regenerateCountSI = 0   ----------------------->   c
       regenerateFlagSI = false    ----------------------->   c
       regenerateListSI = []   ----------------------> c

      for j in secInstances[classes[i]]  ----------------------> cc+c
            availableSlots = []  ----------------------->  cc

            for day in givenSlots  ----------------------> dcc+cc
                  let declare an empty data structure daySlots  ----------------------> dcc
                  for slot in givenSlots[day] ----------------------> dcc*m + dcc


 /*As we know the probability of the running of this code block is very low because it only run when
need to generate whole timetable from the start. So using Probabilistic analysis we found out that
running time of inner code is assumed 1 using probabilistic analysis So we will not need to calculate
the running cost of each line in this block.*/


                        if regenerateFlagSI is true then make slot flag true  ------------> dccm * 1
                              for a in regenerateListSI.slot
                                    make dumFlag false
                                    for b in slot
                                          if slot at b is equal to regenerateListSI.slot at a then
make dumFlag true & break
```

```
                    if dum flag is false then make slot flag false and break

                if( ( (!slotFlag) || (day != regenerateListSI.day)) &&
(teacherTT[secInstances[classes[i]][j].teacher][day][slot] == 0) && (currentTT[day][slot] == 0))
                    then add slot to daySlots;
                regenerateFlagSI = false
            else if regenerateFlagSec is true then make slot flag true
                for a in regenerateListSI.slot
                    dumFlag = false
                    for b in slot
                        if(slot[b] == regenerateListSI.slot[a]) then make dumpFlag True & break
                    if dumFlag is false then make slotFlag = true & break
                                    if(( (!slotFlag) || (day != regenerateListSec.day))
&&(teacherTT[secInstances[classes[i]][j].teacher][day][slot] == 0) &&  (currentTT[day][slot] == 0))
                    then make regenerateFlagSec  false
                                elseif((teacherTT[secInstances[classes[i]][j].teacher][day][slot] == 0) &&
(currentTT[day][slot] == 0))
                add slot to daySlots and then add daySlots to availableSlots;
        eachDay = secInstances[classes[i]][j].numLectures / numDays ----------------------> cc
        extraDays = secInstances[classes[i]][j].numLectures % numDays  ----------------------> cc
        for i in range(numDays)  ----------------------> dcc+cc
                    if there exits an extra day then add eachDay and plus 1 day to count and
decrement extraDays by 1 & repeat ----------------------> dcc+dcc+dcc
            else make count equal to each day ----------------------> dcc
            flag = true , radCount = 0    ----------------------> dcc+dcc


  /*As we know the probability of the running of the three nested loops inside the while is very low, So
using Probabilistic analysis we found out that running time of while loop will be 1 as the probability is
relatively very higher that loop will only execute in each iteration of outer for loop.*/


                    while(flag)  ----------------------> dcc * 1
   {We did not computed the time complexity of thiscode block as it is very that our algorithm, will
execute this code. So we found the cost by probabilistic analysis for this code block}

            if(impossible ||  notPossible || regenerateFlagSec || regenerateFlagSI) then break--> dcc+dcc
        if( impossible ||  notPossible || regenerateFlagSec) then break ---------------------->cc+cc
        if( (!impossible) && (!regenerateFlagSec) && (!notPossible))  ----------------------> c
        add currentTT to TT ----------------------->  c
        secTT[classes[i]] = currentTT ----------------------->  c
        if notPossible is true then make it false   ----------------------->  c+c
        if impossible is true then return("Could not generate in this case, please refresh/restart") --> c+c
    return TT    ----------------------->   1
}
```

```
ranD (slots, count) {
   let i, viableDays, slot ----------> 1+1+1
   for i in slots  ----------> d+1
         if slots[i].length >= count  ----------> d
                  add i in viable days  ----------> d
   if viableDays is empty then return null  ----------> 1+1
   let buff = crypto.randomBytes(2);  ----------> 1
   let n = parseInt(buff.toString('hex'),16)  ----------> 1
   let index = n % (viableDays.length);   ----------> 1
   let day = viableDays[index]    ----------> 1
   for i = 0 to count {    ----------> d+1
         buff = crypto.randomBytes(2);   ----------> d
         n = parseInt(buff.toString('hex'),16)  ----------> d
         let s = n % slots[day].length;  ----------> d
         slot.push(slots[day][s]);  ----------> d
         slots[day].splice(s,1);  ----------> d
   }
   return day, slot  ----------> 1
}
```

Time Complexity Calculations

In the above block we have calculated the running costs for each line and now we will sum all the costs for every line to find the total complexity of algorithm.

T(n) = 1 + d+1 + d + d c+1 nc+c + (n)(c)+nc + nc + nc + nc + 1 + 1 + 1 + c+1 + c+c+c + c + c + c + c + c + cc+c + cc + dcc+cc + dcc + dccm+dcc + dccm + cc + cc + dcc+cc + dcc + dcc+dcc + dcc + dcc+dcc + cc+cc + c + c + c c+c + c+c + 1+1+1 + d+1 + d + d + 1+1 + 1 + 1 + 1 + 1 + d+1 + d + d + d + d + d + 1
After further simplifying these numbers, we have
T(n) = 2d(c^2)m + 10d(c^2) + 6nc + 7(c^2) + 19c + 12d + 19
Now we will choose the polynomial of highest order and after droping the coeficients we will get the desired time complexity for our algorithm which is
T(n) = O(d(c^2)m)

Algorithm Correctness

It is very difficult to prove that our algorithm generates a correct and optimal solution as time table generating problem is one of the NP-Hard problems and as we are using Heuristic approach in our algorithm it becomes even more difficult to prove the correctness of our algorithm. NP-Hard problems

gives us the solution which is nearest to the best possible solution. So in light of all these factors, it was pretty much difficult to prove it. We came up with Inductive Hypothesis to prove the correctness bt we are not sure that it is possible or not.

- **Inductive Hypothesis**
  After k'th iteration, our algorithm will generate a timetable for section[k] which is expected to be without clashes with the other classes.

- **Base Case**
  As k=0, the TT[] will be filled with zero's and there will be no clashes whatsoever as there is no data is placed in the TT[]. So we can say that our inductive hypothesis holds for k=0.

- **Inductive Step**
  It is our to do step to prove that our algorithm generates a clash free Time table for section[k] after k'th iteration.

  Let's say we have input Instances = [{ teacher: "Samyan", sections: ["A"], subject: "AOA", numLectures: "3", numLabs: null, }, { teacher: "Samyan", sections: ["B"], subject: "AOA", numLectures: "3", numLabs: null, }, { teacher: "Samyan", sections: ["C"], subject: "AOA", numLectures: "3", numLabs: null, }, { teacher: "Awais Hasan", sections: ["A"], subject: "DBMS", numLectures: "3", numLabs: null, }, { teacher: "Awais Hasan", sections: ["B"], subject: "DBMS", numLectures: "3", numLabs: null, }, { teacher: "Awais Hasan", sections: ["C"], subject: "DBMS", numLectures: "3", numLabs: null, },]

  Given Time SLots = [3, 3, 2, 3, 2] // Each Index Represents a day

  Teachers = ["Samyan", "Awais Hasan"]

  Sections = ["A", "B", "C"]

  Let's say at k=0 we have

  TT = [[[0,0,0] , [0,0,0] , [0,0] , [0,0,0] , [0,0]], //0th index nested array represents Section A time table
  [[0,0,0] , [0,0,0] , [0,0] , [0,0,0] , [0,0]], //1th index nested array represents Section B time table
  [[0,0,0] , [0,0,0] , [0,0] , [0,0,0] , [0,0]]] //2th index nested array represents Section C time table

  Now after dry running our code for k=1 we sould have time table generated for section[k],

  TT = TT = [[[[0,0,["Awais Hasan", "A", "DBMS"]] , [0,0,0] , [0,["Samyan", "A", "AOA"]] , [0,["Samyan", "A", "AOA"],["Awais Hasan", "A", "DBMS"]] ,
  [["Awais Hasan", "A", "DBMS"],["Samyan", "A", "AOA"]]], //0th index nested array represents Section A time table

[[0,0,0] , [0,0,0] , [0,0] , [0,0,0] , [0,0]], //1th index nested array represents Section B time table
[[0,0,0] , [0,0,0] , [0,0] , [0,0,0] , [0,0]]] //2th index nested array represents Section C time table

In k=2th iteration we should have time table for section[k-1] as well as for section[k]

TT = [[[[0,0,["Awais Hasan", "A", "DBMS"]] , [0,0,0] , [0,["Samyan", "A", "AOA"]] , [0,["Samyan", "A", "AOA"],["Awais Hasan", "A", "DBMS"]] ,
[["Awais Hasan", "A", "DBMS"],["Samyan", "A", "AOA"]]], //0th index nested array represents Section A time table
[[["Awais Hasan", "B", "DBMS"],["Samyan", "B", "AOA"],0] , [0,["Samyan", "B", "AOA"],["Awais Hasan", "B", "DBMS"]] , [["Awais Hasan", "B", "DBMS"],0] , [["Samyan", "B", "AOA"],0,0] , [0,0]], //1th index nested array represents Section B time table
[[0,0,0] , [0,0,0] , [0,0] , [0,0,0] , [0,0]]] //2th index nested array represents Section C time table

In k=3rd iteration we should have time table for section[k] as well as for section[k-1] and section[k-2]:
TT = [[[[0,0,["Awais Hasan", "A", "DBMS"]] , [0,0,0] , [0,["Samyan", "A", "AOA"]] , [0,["Samyan", "A", "AOA"],["Awais Hasan", "A", "DBMS"]] ,
[["Awais Hasan", "A", "DBMS"],["Samyan", "A", "AOA"]]], //0th index nested array represents Section A time table
[[["Awais Hasan", "B", "DBMS"],["Samyan", "B", "AOA"],0] , [0,["Samyan", "B", "AOA"],["Awais Hasan", "B", "DBMS"]] , [["Awais Hasan", "B", "DBMS"],0] , [["Samyan", "B", "AOA"],0,0] , [0,0]], //1th index nested array represents Section B time table
[[["Samyan", "C", "AOA"],["Awais Hasan", "C", "DBMS"],0] , [0,0,0] , [0,["Awais Hasan", "C", "DBMS"]] , [["Awais Hasan", "C", "DBMS"],0,
["Samyan", "C", "AOA"]] , [["Samyan", "C", "AOA"],0]]] //2th index nested array represents Section C time table
Now after k iterations we have a time table generated for all the sections in the section Array without any conflict.

- **Conclusion**
  It is proved that our algorithm provides us with a timetable for all the sections in section[] after kth iteration.

Interfaces for your project

1. Welcome Page
   The welcome page has two controls which are signup and login. If someone is using this software for the very first time, he needs to register and signup button will open a window where he can register him or her. But if user has already registered himself/herself or used this software before, he simply needs to click on the login button which will lead him/her to a window where he/she need to put the credentials.
   - Signup: A user should use this to register himself/herself.
   - Login: A user should use this to login in to system.

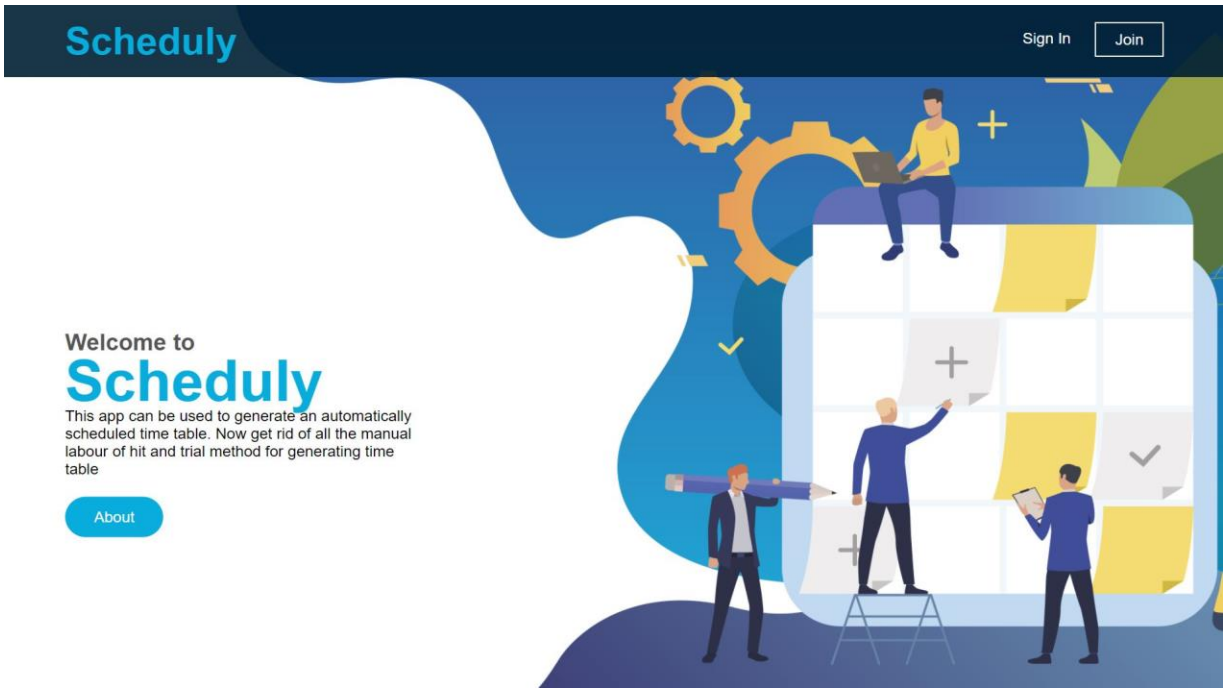

Figure 1: Welcome Page Interface

2. Signup
   Whenever a new user wants to use this system, he must register. Sign up page consists of 5 input boxes and a submit button.
   - Login: To log in to the application.
   - Signup: To register to use this software.
   - First Name: A user should fill this input box by providing his/her first name.
   - Last Name: A user should fill this input box by providing his/her last name.
   - Email Address: A user should fill this input box by providing his/her email.
   - Password: A user should fill this input box by providing a strong password. This password will be re required when user wants to login into the application.
   - Confirm Password: A user should fill this input box by re writing his/her password.
   - Sign Up Button (Right below input boxes): This button is used to submit the credentials.

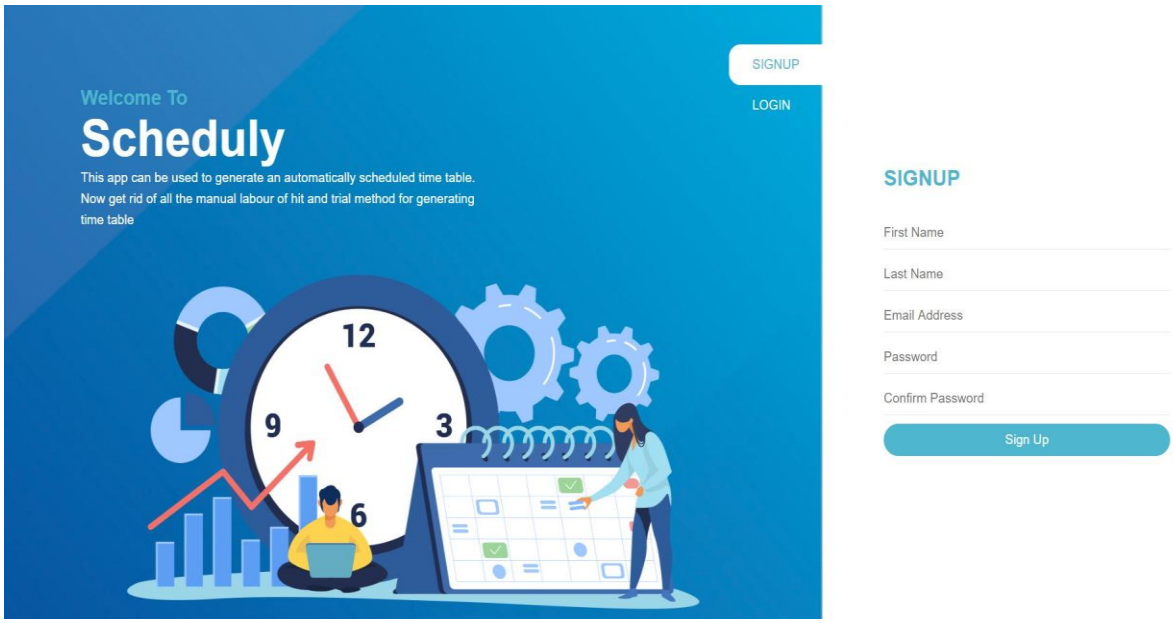

Figure 2: Sign-up page Interface

3. Log In

This page consists of two input boxes and a login button. If the credentials are correct, you will be led to dashboard.

- Email Box: A user should be required to fill this input box with a valid email which he used to register him/her self.
- Password: User should enter his password in this box.
- Login button: By clicking on this button user will be directed to the dashboard if the credentials are turned out to be true.

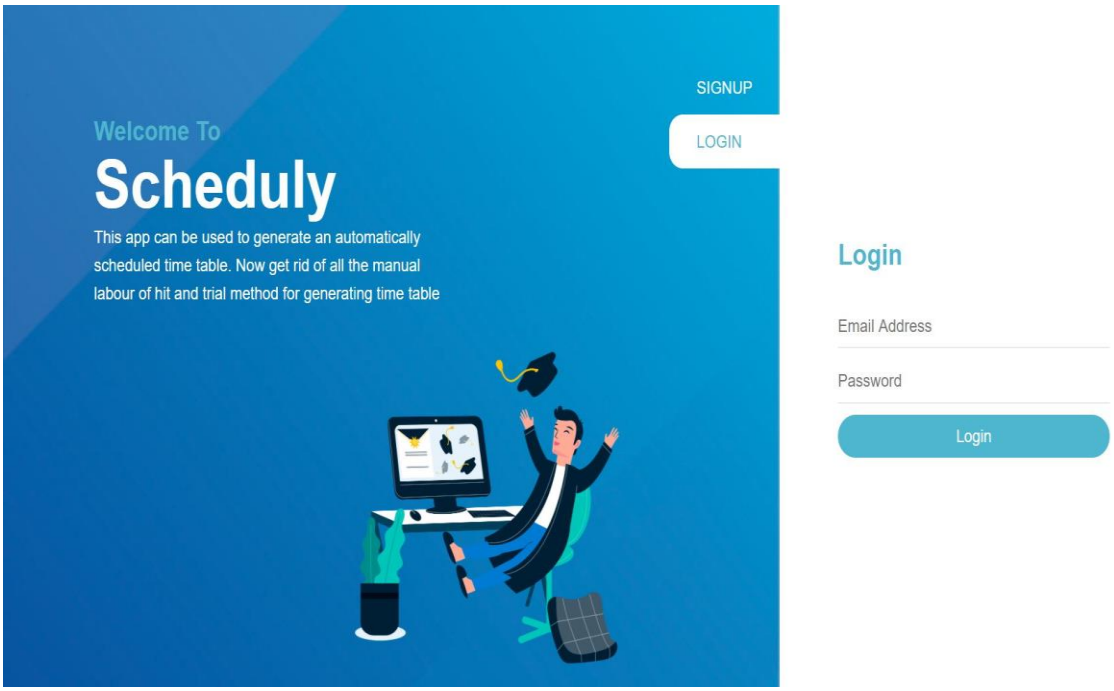

Figure 3: Log-in Page Interface

4. Dashboard

This page will be shown after a user successfully login to the application. It consists of a header and a navigation panel.
In header

- Home Button: This will lead to the dashboard whenever someone presses it.
- Logout Button: This will log out a user from the application and takes him/her to welcome screen.

In navigation panel, there are 5 buttons which are described below.

- Classes: It has a submenu which shows Add and All Classes Button. By clicking the Add button, a new page will be opened where user can add a new class. By clicking on the All Classes button, all the classes will be shown on the UI.
- Subjects: It has a submenu which shows Add and All Subjects Button. By clicking the add button, a new page will be opened where user can add a new subject. By clicking on the All Subjects button, all the subjects will be shown on the UI.
- Teachers: It has a submenu which shows Add and All Teachers Button. By clicking the add button, a new page will be opened where you can a new teacher. By clicking on the All Teachers button, all the teachers will be shown on the UI.
- Slots: It has a submenu which shows Add and All Slots Button. By clicking the Add button, a new page will be opened where you can add a new slot. By clicking on the All Slots button, all the slots will be shown on the UI.
- Generate: This button will send a request to the backend to run algorithm and will show the returned output.

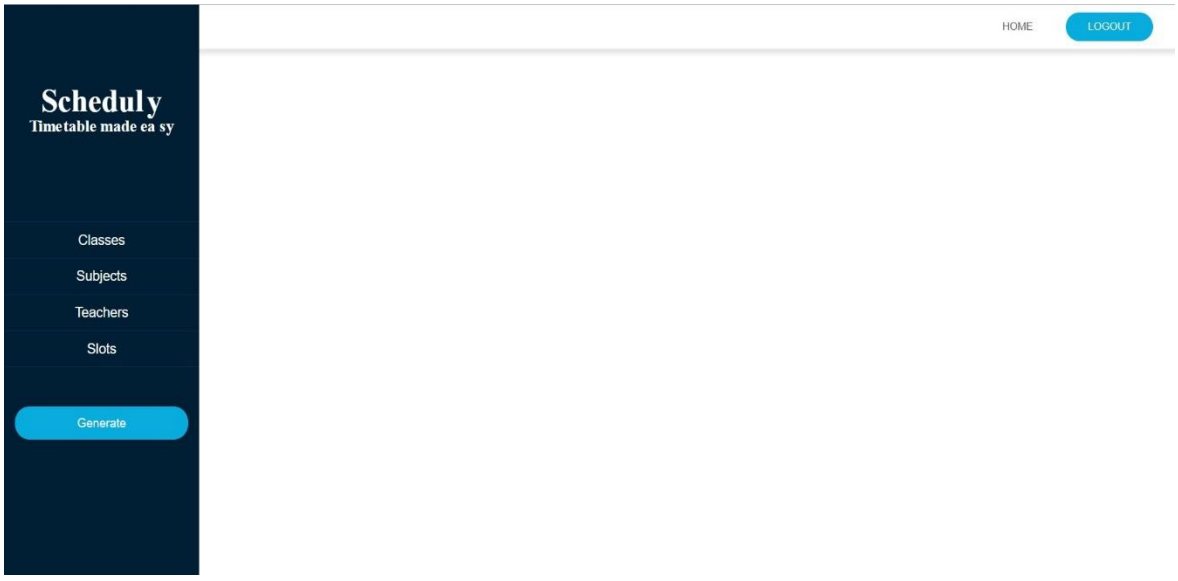

Figure 4: Dashboard Page Interface

5. Add Class

To add a new class, you need to click on classes in navigation panel and select add from submenu. On this page user will be able to add a new class. It has 3 input fields and a submission button.

- Class Name: User need to add a class name such as N-7, N-4.
- Session: This input field requires a session name such as 2018, 2020.
- Section: User need to enter the section of the class.
- Add Class Button: This button will save the records.

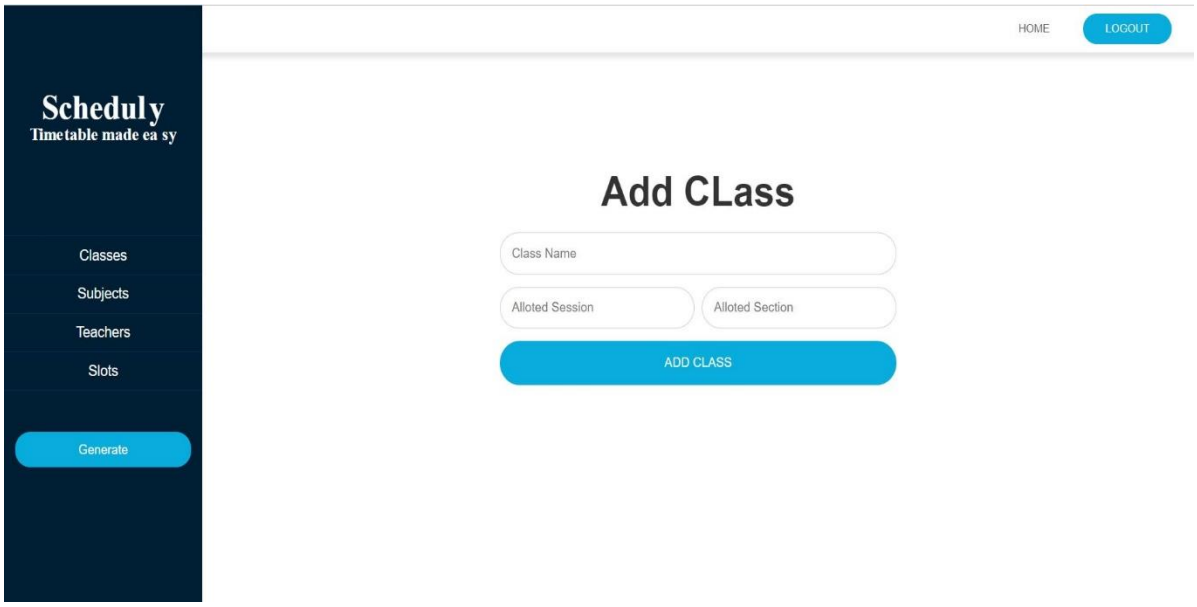

Figure 5: Add Class Page Interface

6. View All Classes

This will show all the classes added by a user. You can go to this page by clicking on classes in navigation panel and select all classes from submenu. It has only one button Remove which will remove a class.

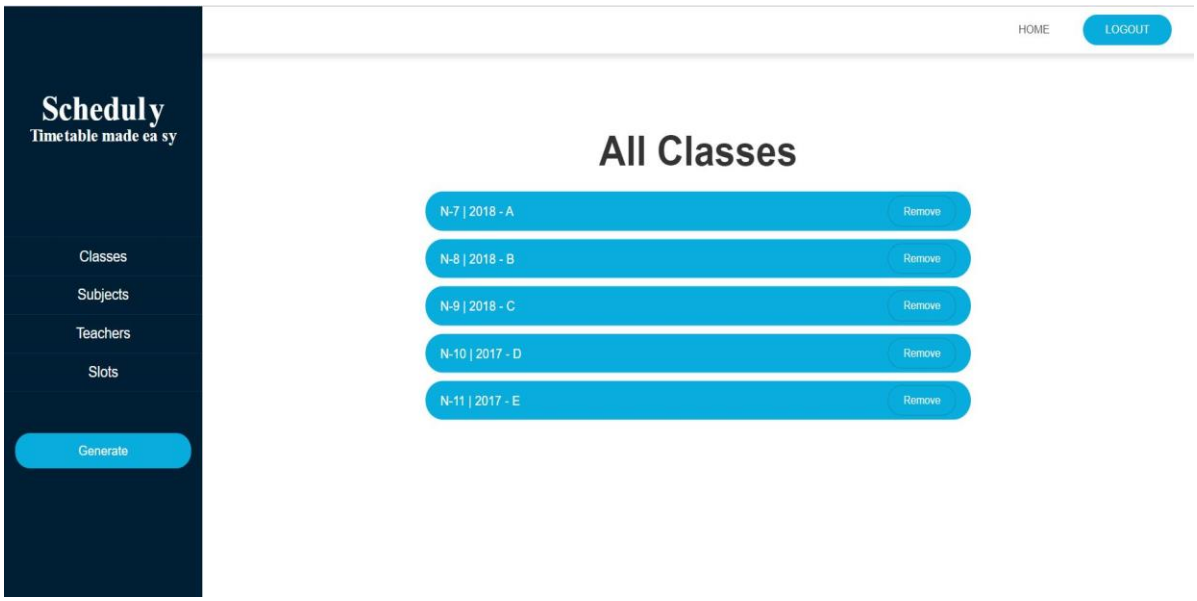

Figure 6: View Classes Page Interface

7. Add Subject

To add a new subject, you need to click on click on subjects in navigation panel and select add from submenu. On this page user will be able to add a new Subject. It has 5 input fields and a add button.

- Subject Name: This will be the name of subject to be added such as Operating System etc.
- Subject Code: This will be the code of subject to be added such as CS311 etc.
- Credit Hours: This will be the credit hours of a subject and it cannot be greater than 3.
- Contact Hours: This will be the contact hours of a subject and it cannot be greater than credit hours.
- Labs: Number of labs assigned to a subject.
- Add Subject Button: This button will save the records.

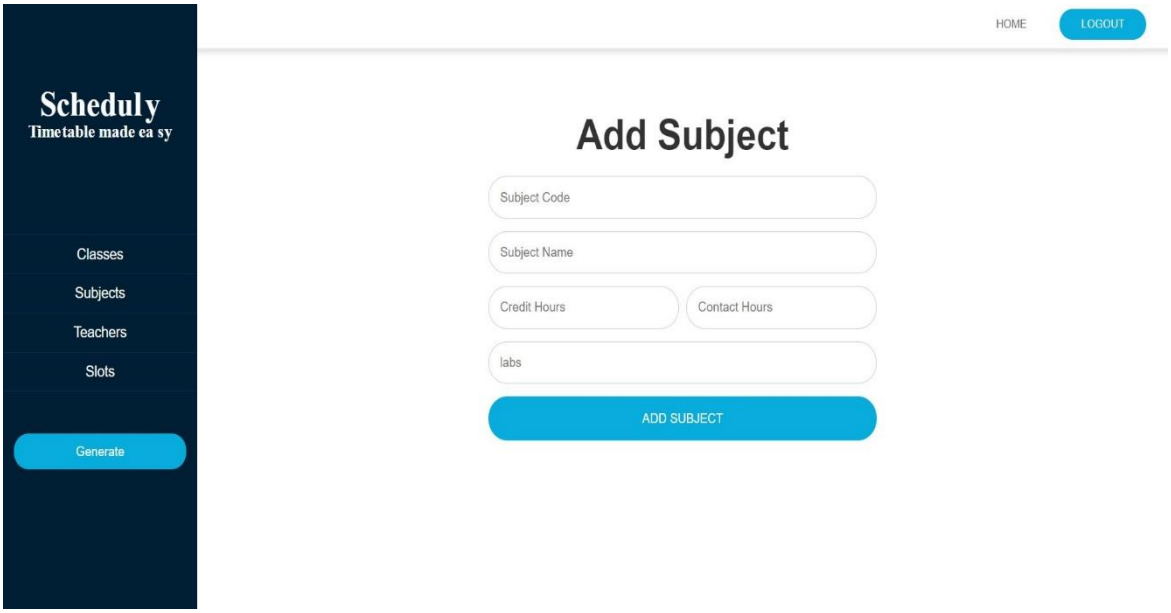

Figure 7: Add Subject Page Interface

8. View All Subjects

This will show all the subjects added by a user. You can go to this page by clicking on subjects in navigation panel on left and select all subjects from submenu. It has only one button Remove which will remove a subject.

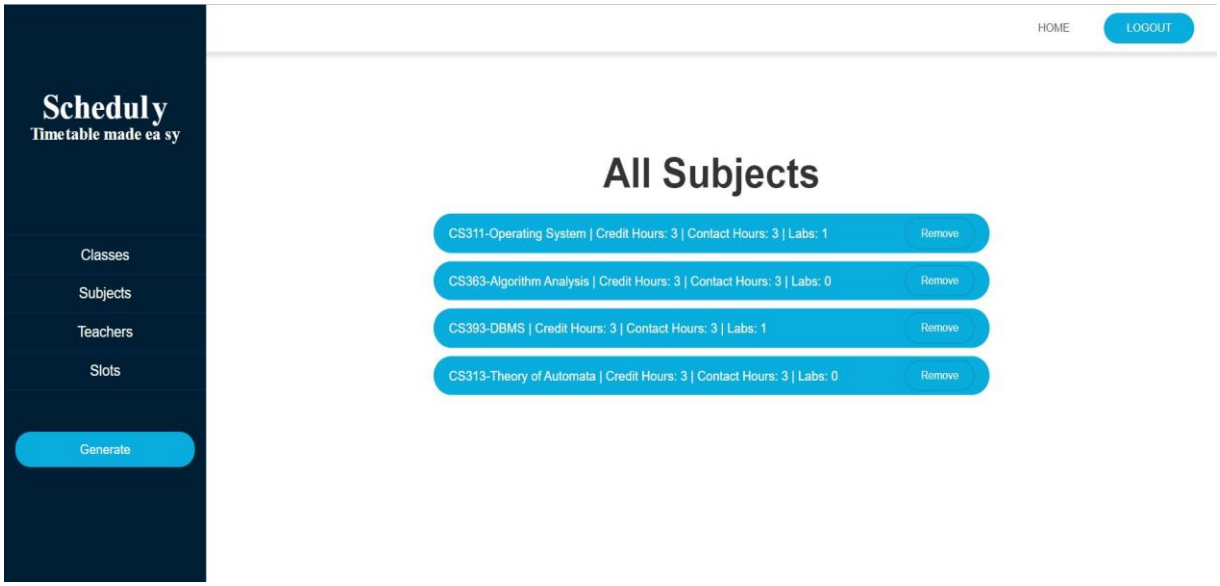

Figure 8: View Subject Page Interface

9. Add Teacher

To add a new teacher, you need to click on click on teachers in navigation panel and select add from submenu. On this page user will be able to add a new teacher. It has 4 input fields and a add button.

- First Name: This will be the first name of teacher to be added such as Samyan.
- Last Name: This will be the last name of teacher to be added such as Qayyum.
- Subject Code: This will be the code of subject to be added such as CS311 etc.
- Reg Number: This will be the Reg Number of a teacher and it cannot be greater than duplicated.
- Working Hours: This will be the working hours of a teacher in a week.
- Add Teacher Button: This button will save the records.

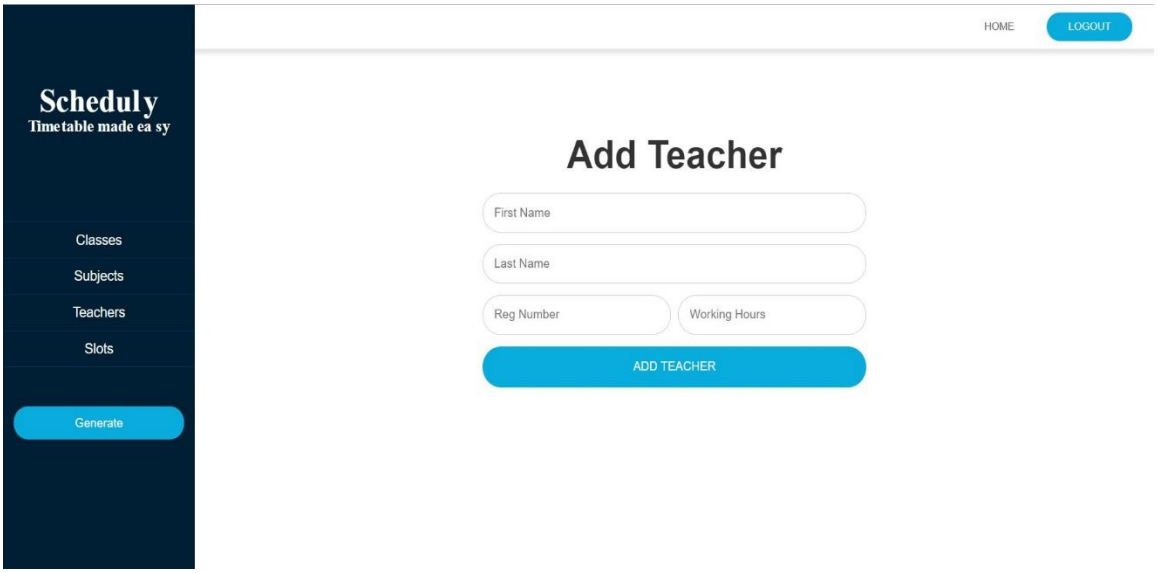

Figure 9: Add Teacher Page Interface

10. View All Teachers
    This will show all the teachers added by a user. You can go to this page by clicking on teachers in navigation panel on left and select all teachers from submenu. It has only one button named as Remove which will remove a teacher.

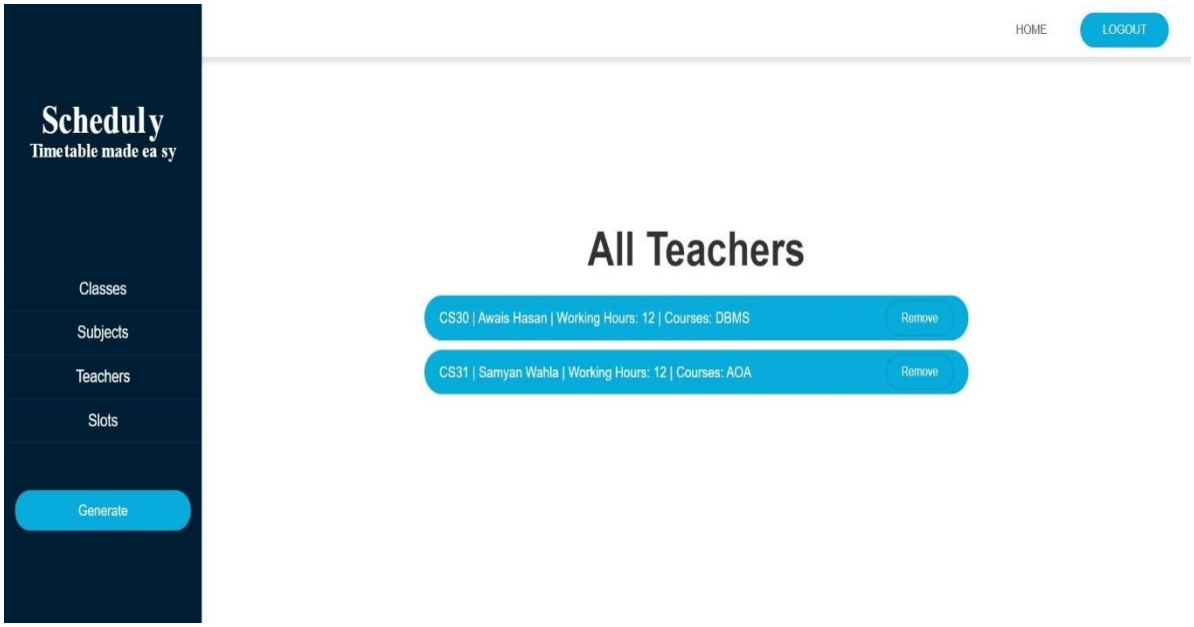

Figure 10: View Teachers Page Interface

11. Add Subject

To add a new subject, you need to click on click on subjects in navigation panel and select add from submenu. On this page user will be able to add a new Subject. It has 5 input fields and a add button.

- Teacher Name: This will be the name of teacher to be assigned to a subject.
- Subject Name: This will be the name of subject to be assigned to the above teacher.
- Session: This input field will be the session name of the class such as 2018, 2020.
- Section: This input field will be the section of the class to which the teacher is being assigned.
- Add Slots Button: This button will save the records.

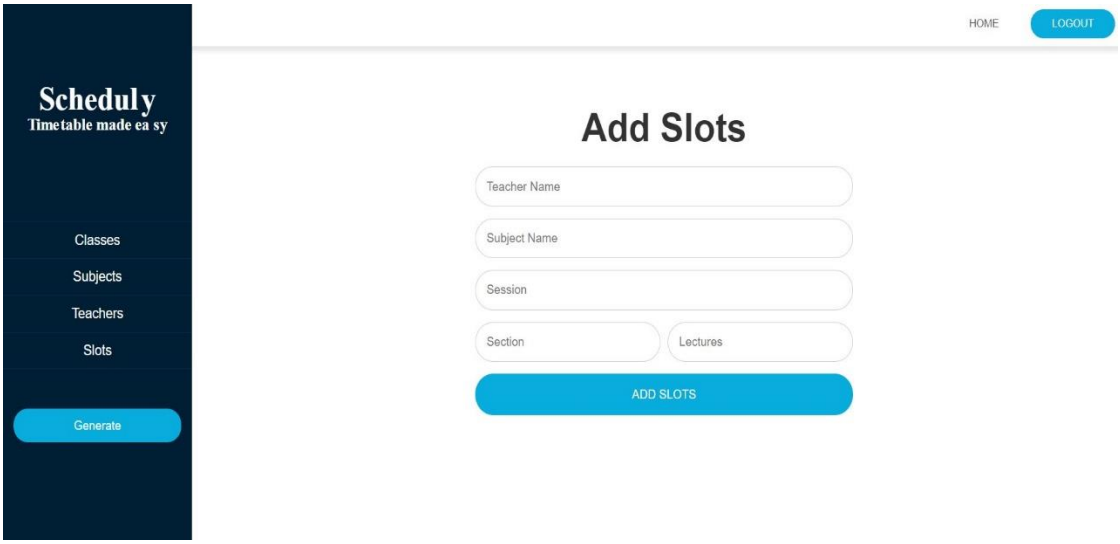

Figure 11: Add Slots/Lectures Page Interface

12. View All Slots

This will show all the slots added by a user. You can go to this page by clicking on slots in navigation panel on left and select all slots from submenu. It has only one button named as Remove which will remove an instance of a slot.

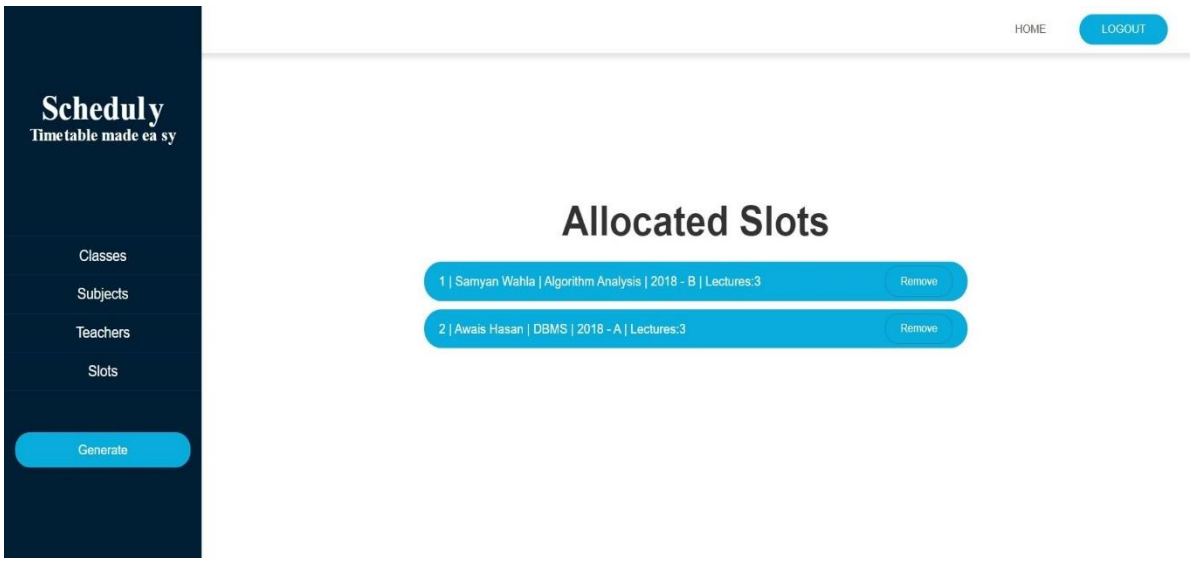

Figure 12: View Slots/Lectures Page Interface

13. Timetable

By clicking on the Generate Button from navigation panel on left, you will see the generated time table. Our output will be shown in tables on this page. Each table will represent time table of one class. Each row will represent a working day and each slot will represent a time slot. In each cell we will show the assigned teacher name and assigned subject name.

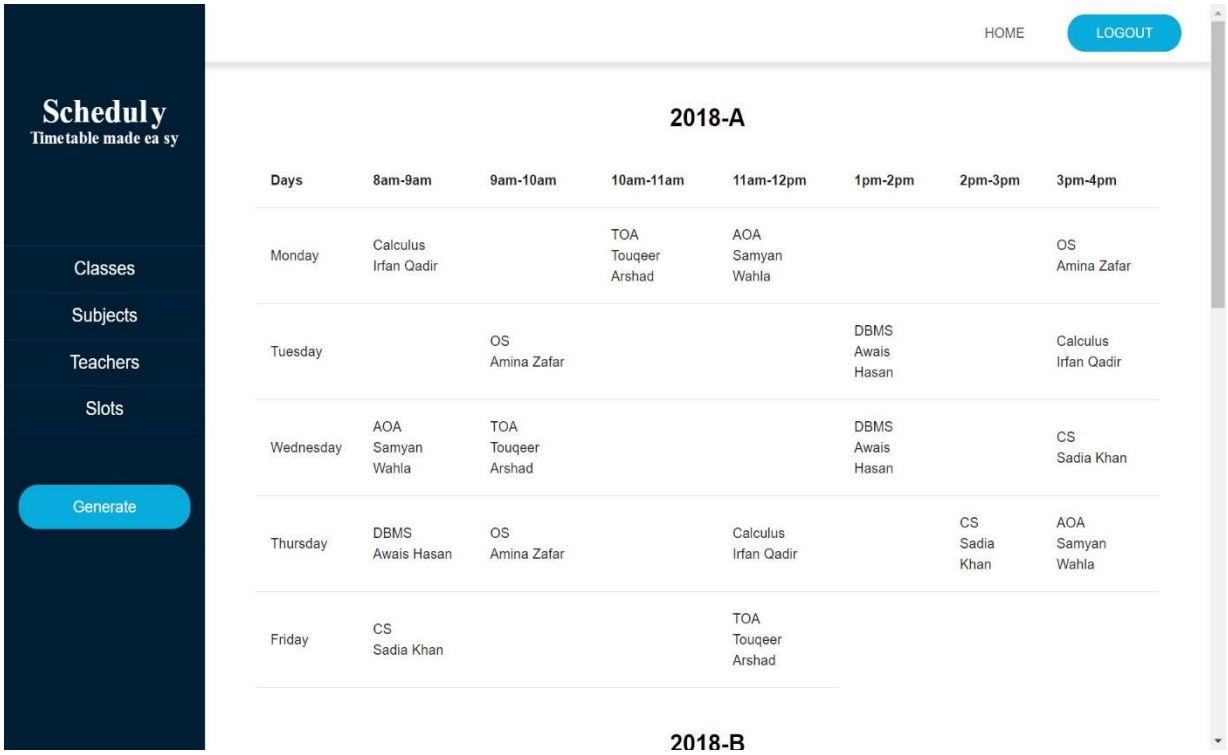

Figure 13: Output page UI

Integration

As such we have not faced any difficulty while integrating the UI with our algorithm on backend. We simply first implemented our algorithm on the backend. Then we designed our frontend without any connections with the backend. After completing the UI, we designed REST API's to handle communicate between UI and our backend. To execute algorithm, we are required to retrieve data from the database, for this purpose we also used a REST API. Using REST API's made it easier for us to handle the data manipulation. Our strategy was to use a REST API which will handle http requests from the UI and will send a response back to UI. Whenever a request is made from the UI using a REST API, the corresponding action to request is performed i.e. the request can be for to save new data, delete some specific record or to generate time table. If the request is to add new data into the database, a post request will be sent to the server which will save the instance using a model (model is already defined on the backend) into the database corresponding to the user ID. If the data is saved is successfully, a response will be sent to the UI. Similarly, for deletion of a record, a delete request will be sent to the server running on backend and data will be removed from database and response will be sent to the UI showing that operation is executed successfully. When a user makes get request to generate a time table, find function will be executed in the API and data will be retrieved from the database and passed to algorithm. The algorithm will return an array of objects which will be sent to the UI. So, all of these requests are handled using an express server on backend and REST API is used to communicate between frontend server and backend server.

Change Requests

Change Request 1: We have added a new UI page to take slots from the user. It has two functionalities add and view. Add page has five input boxes for teacher name, subject assigned to teacher, lectures, section and session respectively. View page has one box for each instance of a slot.

Change Request 2: In our first milestone, we proposed that we will use greedy approach to design our algorithm.

Testing

Testing of our project was assigned to group with group ID CS311-G28 which compromises of two group members with registration numbers 2018-CS-17 and 2018-CS-48. After testing out our project CS311-G28 found the following issues:

Issue #01 Add slots for labs

Testing team proposed the enhancement of handling lab.

Is the issue resolved? If yes then explain how? And if not give the reasoning of not entertaining the issue.

No, the issue is not resolved and closed without working on it.

as it was irrelevant and opened after being told to the testing team that our algorithm does not support labs. Moreover, in our readme file of this project, it was clearly mentioned that labs are not handled in this project. They are being handled just like any other subject. So, we closed this issue without making

any changes to our project.

Issue #02 Interface Improvements

Testing team proposed following improvements in the user interface:

- The layout is not responsive.
- Pages are not automatically being changed. For example, when you sign up or login, the dashboard does not popup.
- You have to manually change the URL.
- The Error messages are not being displayed on the screen.
- Similarly, the success message is not popping up after submitting forms.
- The default URL does not show anything.
- The dashboard page does not show anything by default.
- There is no instruction manual on how to operate dashboard.

Is the issue resolved? If yes then explain how? And if not give the reasoning of not entertaining the issue.

Yes, all the above-mentioned issues are resolved.

The layout was responsive at the time of testing. There was only a slight problem in the view functionality for all the save records in database and it was handled by setting the position of output view boxes. Now there is no such issue about responsiveness exists.

Second problem of not changing was occurred as routing was not implemented at the time of testing. So at the time of testing user needs to change the URL by themselves. This problem was resolved by implementing the routes for all the pages. Problem of changing URL manually also got ridden of after the implementation of the routes.

Error and success messages were not shown on the user interface. It was being checked on the console whether a request is successful or not. But to make our UI more user friendly we implemented the functionality of showing errors on UI.

Default URL does not show anything. This problem also aroused as we had not implemented the routes back then. Now after implementation of routes this problem was also taken care of and a home page is displayed on default URL.

Issue about dashboard is invalid as dashboard URL displays a page from where you can select what you want to do i.e. want to add subjects, add teachers etc.

From our user interface it was pretty clear about how to operate and which inputs are needed to be inserted. But considering that testing team found it hard to operate the software. We have written a user manual on how to insert inputs. It can be found in the documentation folder in the project repository.

Technology	
Programming Language	JavaScript

	React.js for frontend. Node.js for backend. MongoDB as a local database.
Platform	Web Application