**Analysis of Algorithms**

Spring 2020

**Members Details**

| Group ID | **CS311-G54** |
| --- | --- |
| Registration Number of Group Members | **2018-CS-81**  **2018-CS-88** |
| Section | **B** |

**Project Details**

|  |  |
| --- | --- |
| ***Project*** |  |
| Project Title | **Activity Scheduling Tool** |
| Executive Summary | This project is about scheduling the activities. It generates a time table when we input some values in it.  Our project has following specifications.   1. **Authentication:** First of all, the login and registration process which is the essential process of all the projects. A user can login with email and password. 2. **Registration:** If a user has not account then he/she can register himself/herself with username, email and password. An extra thing is that when a user register a token is assigned to himself/herself. **Jwt (Json Web Token)** is used for this process of tokenization a user. 3. **Authorization:** A user when register himself/herself. Then a token from **Jwt (Json Web Token)** Node package is assigned with this user. This token is then used for authorization process. The advantage of this process is that only authenticated user can add, delete records, generate timetable and can show records and also an authenticated user can access routes. The **Routes are also private** in this project which is also due to tokenization of users. 4. **Password Encryption:** because security is very important for any project. And a good programmer cannot compromise on security while he/she building a project. So in this project we used **Bcrypt** Node package to encrypt our user password. So when a user register himself/herself. Password will be stored in the encryption form in the MongoDB database and at the time of login, the user input password will be encrypted first and then match with the password stored in the database. 5. **Adding Records:** After the login process, we can add records. We can add list of rooms, list of course with instructor name and credit hours and daily working hours. The above added record will directly be added in the MongoDB database. 6. **Show All Records:** We can show all our added records which is in the database. 7. **Delete Records:**  we can delete the records one by one which is stored in the database so that we can again add our records and can use those records to generate a new table. 8. **Generate Timetable:** This is the main section of our project. In this section when we click on the generate timetable button we will generate a timetable which is based on the inputs stored in our database by the user. |
| ***Business Case*** |  |
| Outline the business need for the project | From business point of view most of the institutions find it diffult to manually arrange the activities using minimum number of resources and it is time consuming too. So our web application will resolve all these issues inn minimum span of time. |
| End user of the product | * Universities/school/college for scheduling classes or labs. * Hotels which have a lot of rooms and charges on the basis of hours. * Banquet halls having considerable amount of rooms. |
| Motivation for Project | The motivation for this project was it was really interesting although a bit difficult because we devised our own algorithm which took a lot of time but we learnt a lot of this and secondly we don’t think that we can learn more than this in any other project offered. A side reason is also that if our project is used to help others it will be great. |
| Description of the project objective(s) | The key objective of our project was to generate a time table by using the information which end user will enter. It also ensures the security of user because we have pretty secure authentication procedure and encryption as well. |
| State the level of impact expected should the project proceed and implications of not proceeding | Yeah the implementation had an impact on both operational level and strategic level because we went for implementation most of the times it did worked so first we have to change the strategy and then the operation. So it has an impact on both levels. |
| Functional Requirements | * The user can login or signup. * Password Encryption * Adding Records * Show All Records * Delete Records: * Generate Timetable |
| ***Benefits*** |  |
| What benefits are expected/ anticipated? | * It will save a lot of time to make different strategies to schedule activities. * Management authorities can use this application to schedule meetings/classes etc. * Hotels/Banquet or any other person or organization having a lot of rooms and want to schedule some activities simultaneously without any headache of designing or drawing manually can use this. |
| ***Implementation Details*** |  |
| Link to Github Repository | <https://github.com/Dawoodjaveed/CS311S20PID54> |
| Total Number of commits in repository before 5th Aug 2020 | 27 commits |
| Exact contribution of each member | 2018-CS-88(20 commits)  2018-CS-81(7 commits) |
| ***Commits in github repository by each member*** | |
| |  |  | | --- | --- | | **Member Registration No.** | **Total Commits** | | 2018-CS-88 | 20 | | 2018-CS-81 | 7 | |  |  | | |
| **Details of commits** | |
| |  |  |  |  | | --- | --- | --- | --- | | **Sr. No.** | **Details of commit** | **Date** | **Member Reg No.** | | 1 | [2018-CS-88 first commit](https://github.com/Dawoodjaveed/CS311S20PID54/commit/77b58e5c21f0c12daee598613993f1ca44799e79) | Jul 4, 2020 | 2018-CS-88 | | 2 | [this is a first react initialize project wiht dashboard](https://github.com/Dawoodjaveed/CS311S20PID54/commit/86ef503f9061ba7c70bebe1b90d0be3bbdbab90c) | Jul 26, 2020 | 2018-CS-88 | | 3 | [Adding Records Component](https://github.com/Dawoodjaveed/CS311S20PID54/commit/e5a4afe085669514fdcc8ef5f9eac6d78f4e7be9) | Jul 26, 2020 | 2018-CS-88 | | 4 | [Deleting And Showing Records Component](https://github.com/Dawoodjaveed/CS311S20PID54/commit/d28ee3b8ee6be84355426df407128a9baa7d5430) | Jul 26, 2020 | 2018-CS-88 | | 5 | [Setup for Private Routes](https://github.com/Dawoodjaveed/CS311S20PID54/commit/50f6d14411027ba766022e6e40b865b42471f78a) | Jul 26, 2020 | 2018-CS-88 | | 6 | [Some Addintional Assests](https://github.com/Dawoodjaveed/CS311S20PID54/commit/c002ac03249034b261ecc0449c9d7100450e29eb) | Jul 26, 2020 | 2018-CS-88 | | 7 | [React Redux Store With Actions,Reducers](https://github.com/Dawoodjaveed/CS311S20PID54/commit/94704170ac6f961c9e6a2764716be4635f22bb37) | Jul 26, 2020 | 2018-CS-88 | | 8 | [User Full Auth Component](https://github.com/Dawoodjaveed/CS311S20PID54/commit/91dfc0113831e15cc0669da31ac2394cc49ab43a) | Jul 26, 2020 | 2018-CS-88 | | 9 | [Timetable Component](https://github.com/Dawoodjaveed/CS311S20PID54/commit/c44782bd3802ec2443bcd9d314fe69cef569ed21) | Jul 26, 2020 | 2018-CS-88 | | 10 | [Algorithm](https://github.com/Dawoodjaveed/CS311S20PID54/commit/bded1bb6765de1eb3de1f4999af4074613d2d672) | Jul 26, 2020 | 2018-CS-88 | | 11 | [All Other Static Files](https://github.com/Dawoodjaveed/CS311S20PID54/commit/8fdb1af6ab20c4ee2def39f0814546f2007550e3) | Jul 26, 2020 | 2018-CS-88 | | 12 | [Improvements](https://github.com/Dawoodjaveed/CS311S20PID54/commit/64201a08948b162f8a9f72a131402d52fc164ce4) | Jul 26, 2020 | 2018-CS-88 | | 13 | [ReImplementataion of Algorithm](https://github.com/Dawoodjaveed/CS311S20PID54/commit/a831db6f67dc2fa2cd2a1c3edd5ac706f5c8ed52) | Jul 26, 2020 | 2018-CS-88 | | 14 | [Update Pseudo Code](https://github.com/Dawoodjaveed/CS311S20PID54/commit/1be2589fa587d0516bf131a535e021e700871996) | Jul 26, 2020 | 2018-CS-88 | | 15 | [improvments in TimeTable UI](https://github.com/Dawoodjaveed/CS311S20PID54/commit/7eaeb2cf9eae8896370bea8fac10f16906dc5aad) | Jul 26, 2020 | 2018-CS-88 | | 16 | [Clinet Side Modification](https://github.com/Dawoodjaveed/CS311S20PID54/commit/ac387edadedecaebfea5a0570a69c3104b93dd86) | Jul 28, 2020 | 2018-CS-88 | | 17 | [changin allRecords table ui](https://github.com/Dawoodjaveed/CS311S20PID54/commit/793875063ce6b8306d28702cda0f6de5bf1d6fde) | Jul 28, 2020 | 2018-CS-88 | | 18 | [Ui Imrovements and validations](https://github.com/Dawoodjaveed/CS311S20PID54/commit/d5bee1a5c005abcdacc3934a269c33b890ce2b90) | Jul 29, 2020 | 2018-CS-88 | | 19 | [Static Files](https://github.com/Dawoodjaveed/CS311S20PID54/commit/a0c3b7dd1e1c4cfa2567bb1fa013c68ba6a9f431) | Jul 26, 2020 | 2018-CS-81 | | 20 | [All MongoDb Data Models](https://github.com/Dawoodjaveed/CS311S20PID54/commit/e455cf6a04f3ca51996bc7d46c9b7dcece709b62) | Jul 26, 2020 | 2018-CS-81 | | 21 | [Config File](https://github.com/Dawoodjaveed/CS311S20PID54/commit/9702568d4bc3749d876473f2db988757c13c4a1b) | Jul 26, 2020 | 2018-CS-81 | | 22 | [Add files via upload](https://github.com/Dawoodjaveed/CS311S20PID54/commit/407f3a994cdd566c439a59bf012fff6a2205874f) | Jul 26, 2020 | 2018-CS-81 | | |
| Have you used built in algorithms or you have implemented yourself? | We have tried a lot of built in algorithms like genetic algorithm but we were unable to implement so yes we have at last implemented our own algorithm which took some serious time and hard work. But we did it. |
| Formats of input | Our inputs are as follows.   1. Course Name(string) 2. Course Credit Hours(integer) 3. Instructor Name(string) 4. Room Number(string) 5. Daily Working Hour(integer)   Our input format will be as follows that we input names of courses along with their credit hours and the instructors who will teach this course. And input Daily working hours and a list of room numbers in which theses course can teach. |
| Validations | **Input Validations:** There are some input validations in our project.   1. The below picture shows the validation of adding a room if a room is already present in the database then it will not add it in the database because if a duplicate room will be present then number of conflicts among classes will never be zero and we cannot schedule our timetable.      1. The below picture shows the validation of adding a course if a course is already present in the database then it will not add it in the database because if a duplicate course then again the same case as for duplicate rooms.      1. The below picture shows the validation of adding Daily Working Hours and it means that we can add it only for once and there is no need to add it for second time. 2. This validation is for all inputs and if you try to submit any form with empty value then it will not be possible for you. |
| Format of output | After processing all the inputs our algorithm will generate a schedule (here schedule means timetable) which consist of a list of classes. Each class has a instructors name, course name, timeslot, day and a room number. |
| Deployment | Not yet but will do that before viva. |
| ***Details of algorithms*** | |
| Algorithm Pseudocode //we input names of courses along with their courseCreditHours  //and the instructors who will teach this course. And input Dailyworkinghours  //and a list of room numbers in which theses course can teach  //length of timeslots array depends upon the dailyWorkingHours  timeslots = [  "08:00-09:00",  "09:00-10:00",  "10:00-11:00",  "11:00-12:00",  "01:00-02:00",  ]  days = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday"]  totalCreditHours = sum(creditHours)  totalDailyWorkingHours = dailyWorkingHours\*5  COURSECREDITHOURSDECEREMENTAL(courseName, courseNamesList){  for (m=1 to length of courseNameList) {  if (courseNamesList[m] is equal to courseName) {  if (parseInt(creditHoursList[m]) !== 0) {  creditHoursList[m] = creditHoursList[m] - 1;  break;  }  }  }  };  INITILIZE(roomslist, cNames, cInstructors){  filter = 0;  if (totalCreditHours <= totalDailyWorkingHours) {  timeTable = [];  var index = 0;  for (var i = 0; i < creditHoursList.length; i++) {  for (u=1 to creditHoursList[i]) {  randomRoomNo = randomly selected roomNo from roomslist  randomTimeSlot = randomly selected timeslot from timeslots  randomDay = randomly selected day from days  myclass = {  courseName: cNames[index]),  instructorName: cInstructors[index],  roomNo: randomRoomNo,  timeslot: randomTimeSlot,  day: randomDay,  }  Adding myclass into timeTable list  }  index++;  }  if (checker == 0) {  timeTableValues = all values of timeTable[0] object  Adding timeTable[0] object into conflictsFreeTimeTable  COURSECREDITHOURSDECEREMENTAL(timeTableValues[0], cNames);  checker++;  INITILIZE(roomslist, cNames, cInstructors);  } else {  if (confilictsFreeTimeTable.length !== totalCreditHours) {  for (k=1 to length timeTable) {  timeTableValuess = values of timeTable[k] object  for (g = 0 to length of confilictsFreeTimeTable) {  conflictsFreeTTValues = values of confilictsFreeTimeTable[g] object  if (conflictsFreeTTValues[3] is equal to timeTableValuess[3] and conflictsFreeTTValues[4].localeCompare(timeTableValuess[4]) {  if (conflictsFreeTTValues[2] is not equal to timeTableValuess[2])  {  filter++;  }  } else {  filter++;  }  }  if (filter is equal to length of confilictsFreeTimeTable) {  Adding timeTable[k] to confilictsFreeTimeTabl array  COURSECREDITHOURSDECEREMENTAL(timeTableValuess[0], cNames)  }  filter = 0;  }  INITILIZE(roomslist, cNames, cInstructors);  }  for (k = 1 to length of confilictsFreeTimeTable) {  values = values of confilictsFreeTimeTable[k] object  Adding values[0] in coursesProps array  Adding values[1] in instructorsProps array  Adding values[2] in roomsProps array  Adding values[3] in timeslotsProps array  Adding values[4] in daysProps array    }  }  }  running++;  return confilictsFreeTimeTable  }; Algorithm Working Our algorithm will take all the inputs which are mentioned above in input section and these inputs will be directly fetch from the database. Beside these inputs two other list are also used in this algorithm. These two are the list of timeslots and a list of days. These are the following operations that are take place in the whole algorithm.   1. First of all, it will check either total credit hours of all classes are less than the total Working hours ((daily working hours) \*5). It will check because if we insert subject that have total credit hours more than total working hours than how a time table can be possible? If we have less credit hours, then only possible that all classes can be take place by using these timeslots. 2. After checking this condition, it will generate classes with course name, instructor name and randomly selected the room number, timeslot and day from all arrays. The number of classed that will generate firstly with all these properties depends upon the number of total credit hours. Because course has a number of classes. 3. When all classes are generated then now it is time to select a class form all these generated classes. 4. **Criteria for selecting class:** A class which is selected should have zero conflicts with each and every class which is already in the selected list (a list which is conflicts free list). But what we mean by conflicts free? It means that a class which has a room slot and day that will not match with any other class which is in the already selected list. But if we find a class with timeslot and day matches with already present class then this these two classed should not match the room number. And this is our criteria for selecting a class. 5. **How we will select a class:** First class from generated class will definitely in the selection list and we will continue selecting classes till our number of selecting classes will become equal to total credit hours. For this we will compare our all classes which are in the selection list with the all other generated classes. If we find a class, then we will add it in our selection list but if we not find then we call our function again and again we will search and so on. 6. **Correctness of Algorithm:**      1. Algorithm Complexity: Time complexity of algorithm is n^2 + n | |
| ***Interfaces for your project*** | |
| User Login   Here the user will enter email address and password (which will be encrypted) and login by clicking the LOGIN button. User Registration   If user don’t have an account he/she can sign up by entering the username, email address, password, Re-entering and finally pressing the **SIGN UP** button. Dashboard   In the dashboard user can use add room option to add a room for activity, add course for activity, add daily working hours, view all records, delete a record, generate time table and logout by pressing the respective button in dashboard.  Now These are the following different options which are available for an authenticated user. Adding Room   The user can enter the room no. and then add it by clicking **ADD ROOM** button. Adding Course   The user can add course by entering the course name, credit hours and instructor name in respective box and finally clicking the **ADD COURSE** button. Adding Daily Working Hours   The user can add daily working hours here. Showing All the Record   Here all the inserted records will be displayed. It will be visible in a tabular form after clicking the **All Records** button in dashboard. Delete Record   The course, room or daily working hours can be deleted by hitting respective button here. Generate Timetable   Finally, the table can be generated by clicking the **Generate Timetable** button in dashboard. | |
| ***Integration*** | |
| I integrate the front end which is in reactjs with nodejs. | |
| ***Change Requests*** | |
| *No* | |
| ***Testing:*** | |
| Dear sir, the testing of our project is not done by the other group. So we got no issues in our project. | |
| ***Technology*** |  |
| Programming Language | JavaScript |
| Platform | Web Application |

**CS311 Term Project Configuration Document**

|  |  |
| --- | --- |
| **Project:** | Activity Scheduling |

**Programming Languages, Tools and software(IDEs) used for Project implementation:**

|  |  |  |  |
| --- | --- | --- | --- |
| Sr. No. | Software and Tools | Version | Reason |
| 1 | Visual Studio Code(IDE) | Version 1.47 | For Coding |
| 2 | Postman | Version 7.29.3 | For checking Http Requests functionality |
| 3 | MongoDB Compass Community | Automatically installed with MongoDB | For Database management |
| 4 | Chrome | 84.0.4147.105 | For project testing |
| 5 | Cmd | General | For doing many command line tasks |
| 6 | Node.js | Version 12.18.3 LTS | It is necessary to install. It is used for our project and node packages will be used with this. |
| 7 | MongoDB(Locally) | Version 4.4 | Locally install this Database |
| 8 | GitHub Desktop | Latest version | For dealing with github |

|  |  |  |  |
| --- | --- | --- | --- |
| Sr. No. | JavaScript(Libraries) | Version | Reason |
| 1 | React.js |  | It will be used for front end. It is the front end library of JavaScript. The basic operations performed in this by components. reactjs is based on components |
| 2 | Node.js |  | Node.js is not a library not a framework. Some people confuse it that it is a framework but that is not truth it is only the runtime for JavaScript on the browser and used for server side. |
| 3 | Express.js |  | Express.js, or simply Express, is a web application framework for Node.js, released as free and open-source software under the MIT License. It is designed for building web applications and APIs. It has been called the de facto standard server framework for Node.js. it increases Node.js functionality and make it more easily especially routes in express are much simpler and efficient than in node.js and it has been called standard server framework for node.js |
| 8 | Some Other Node Packages |  | It is some additional packages of node.js. |

**Project Configuration Guidelines (Step by Step):**

# First of all download project from GitHub repository([CS311S20PID54](https://github.com/Dawoodjaveed/CS311S20PID54)).

# Then you have to install all the tools and software that are mentioned above.

# After installing the all above software. Now it is time to first setup the database for our project. For this first of all you have to go to following folder and this folder will be created when you install MongoDB. The folder path is as C:\Program Files\MongoDB\Server\4.2\bin and then open cmd here and type mongod and press Enter. This will run the mongodb server. Now open the MongoDB compass community and make the following database structure.

# Database structure is as follows. Database Name: ActivityScheduling Collections: Users, Rooms, Courses and DailyWorkingHours

# Open the folder which you downloaded from GitHub in the Visual Studio Code.

# Open two terminals in visual studio code or in Cmd.

# In first terminal, go to client folder and type npm i (it will install all the node modules that are required for client side of project). After this type npm start (it will start the client side of project).

# And in the second terminal go to server folder and type npm i (It will install node modules that are required for server side of project) and after this type nodemon index.js (it will start server side of project). congratulations you project is ready