IT314: Software Engineering

Mid-Assessment Documentation

Group-30: Dynamic Timetable System

Group Members:

Margi Hingrajia:	201801014
Priyal Raj:	201801106
Yash Patel:	201801134
Pratik Parmar:	201801211
Harshil Goti:	201801130
Maharshi Vaghela:	201801216
Nishant Shah:	201801403
Jaydeep Machhi:	201801452
Manish Khandar:	201801222
Jaykumar Darji:	201801460

1) Problem Statement

Most colleges have a number of different courses and each course has a number of subjects. Now there are limited faculties, each faculty teaching more than one subject. So now, the time table needed to schedule the faculty at provided time slots in such a way that their timings do not overlap and the timetable makes the best use of all faculty subject demands. We use a genetic algorithm for this purpose. In our Timetable Generation algorithm, we propose to utilize a timetable object. This object comprises Classroom objects and the timetable for every them likewise a fitness score for the timetable. Fitness score relates to the number of crashes the timetable has regarding alternate calendars for different classes. Classroom objects consist of weekly objects. Weekly objects consist of Days. Also, Days comprises Timeslots. Timeslot has an address in which a subject, student gathering going to the address and educator showing the subject is related. Also further on discussing the imperatives, we have utilized composite configuration design, which makes it well extendable to include or uproot as numerous obligations. In every obligation class, the condition as determined in our inquiry is now checked between two timetable objects. On the off chance that condition is fulfilled i.e there is a crash is available then the score is augmented by one.

2) Functional Requirements

A) User Requirements:

Dean :

1. Login/Logout: Dean (admin member) can log in or log out of the system with the set password and email id to access the other features.
2. Edit profile: Can change the personal details like email id, password, address, etc.
3. Add/Remove the faculty members and students: The dean can register new faculty members as well as students.
4. Add/Remove the courses: Only the dean can amend the courses that add new courses or remove the ones no longer continued.
5. Review comments: See and reply to comments

Faculty :

1. Login/Logout: Faculties can log in or logout of the system with the set password and email id to access the other features.
2. Request for the edit to the dean: Faculties can send a message to the dean if they want any changes.
3. View timetable (personalized)
4. View student list of the course
5. Edit profile: Can change the personal details like email id, password, address, etc.
6. Comment (read and write)
7. View free slots: Can view the slots which are free and can be used.
8. Add notes: Write and save whatever is important

Students :

1. Login/Logout: Students can log in or log out of the system with the set password and email id to access the other features.
2. Edit the profile: Can change the personal details like email id, password, address, etc.
3. View timetable according to the current batch they are in.
4. Send messages to faculties
5. Comment (read and write): Can write their doubts as comments if any.
6. Add notes: Write and save whatever is important

B) System Requirements:

a) A stable internet connection: A minimum speed of 512kbps.
b) Supported browsers: As it is a web app, it should work well with browsers like Firefox, Internet Explorer, Google Chrome, etc.

3) Non-Functional Requirements

a) Security: The system should be available on the LAN of the institution only and shouldn't be accessed by anyone from elsewhere. This will ensure the security of all the data(passwords, etc).

b) Performance: The system should be functional and accessible all the time. Also, it should support all the users of the institute at the same time. It should also differentiate well between the registered user, new user and not allow duplicate accounts.

c) Data backup: The system keeps the back-up of all the data from time to time.

d) ***Design:*** As it is a web app, it should work well with browsers like Firefox, Internet Explorer, Google Chrome, etc. Also, the system should be designed using HTML 5, CSS, and Javascript.

4) Elicitation Techniques

We have primarily used four requirement elicitation techniques. A list of them are given below:

(1) Document Analysis: This technique includes the study of already existing systems and documents related to the system which we are going to make. We used the following documents to get the idea of the requirements.

- Dynamically Time Table Generation System Project – 1000 Projects
- Automated college timetable generator
- https://github.com/pranavkhurana/Time-table-scheduler
- https://github.com/NDresevic/timetable-generator
- https://github.com/pranavkhurana/Time-table-scheduler

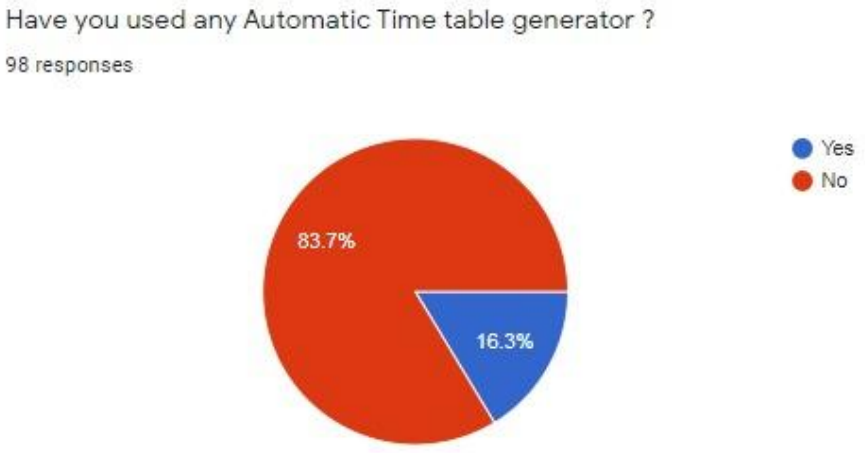
(2) Observation: This technique is useful to collect data just by observing the users and stakeholders who are going to use the system. In our case, users are students and professors. We tried to put ourselves in the place of a student and thought about which functionalities we want in the timetable scheduler. By doing so we realized that as a student I want a load-balanced scheduler, personalized view of timetable, a brief description of the subject, etc.

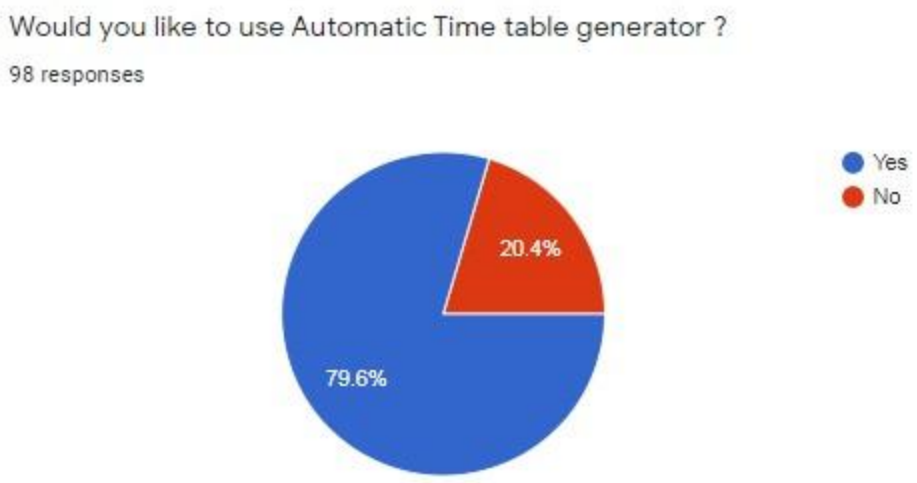
Professor also generally prefers a load distributed schedule, personalized view, the timetable must be dynamic, etc. Since we as business analysts observed users and stakeholders, this method is called Active Observation. Another method is passive observation in which the analyst observes the subject experts.

(3) Brainstorming: In terms of the system which we are going to make, this technique was the most important one to gather the requirements. We regularly arranged meetings with our team members and did brainstorming there. During each session of the brainstorming, we decided on a particular section to discuss. Based on the topic which we are discussing each member tried to give their innovative ideas and pros and cons of that. After one member had proposed his innovative idea, each member of the team gave their opinions regarding this idea and gave some final comments on whether we should include this idea in our

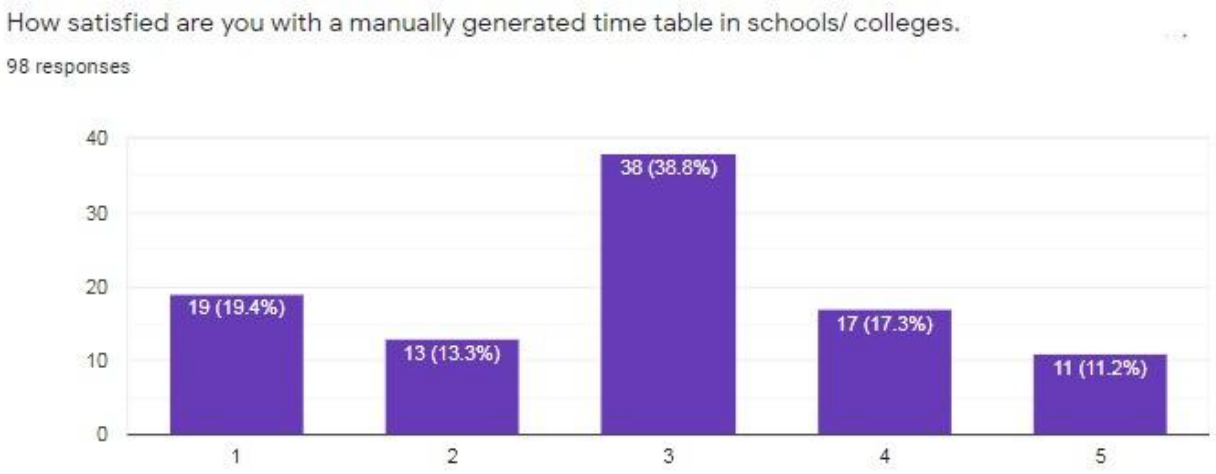
system or not. The final decision about including ideas into the system would be the one to which the majority of the members are agreeing. This has helped a lot in gathering requirements such as while designing the blueprint of the webpages everyone shared their innovative ideas and then we would make very good-looking and easy-to-use blueprints of the web pages.

(4) Surveys / Questionnaires: Our main intention behind this method was to see whether users are interested in our product and to study how to prioritize the requirements of the timetable. For this purpose, we circulated a questionnaire form and analyzed the collected data. We received responses from around 100 students.

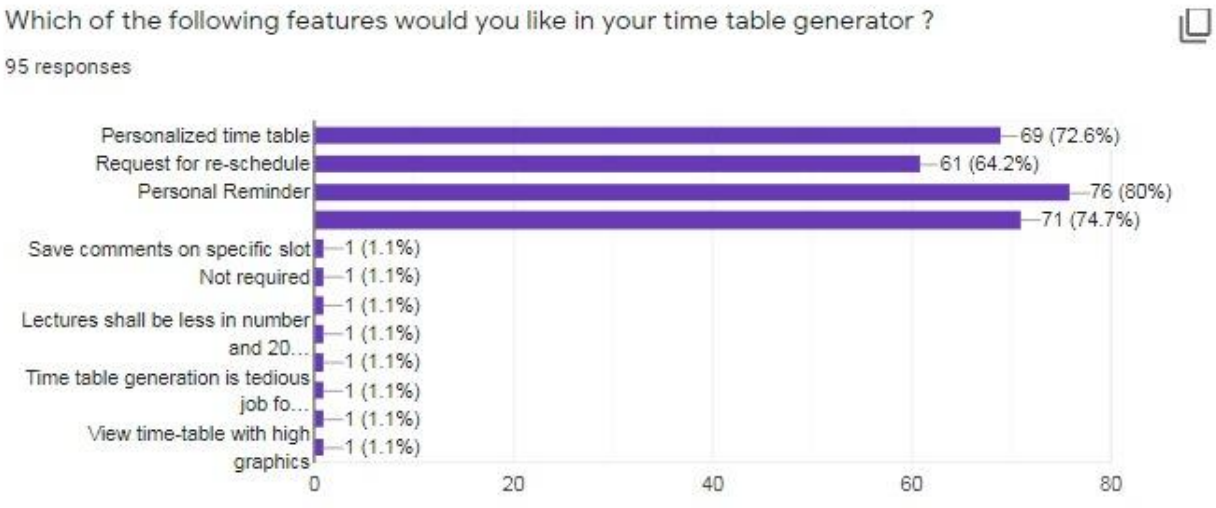




We have observed that 83.4% of the students have not used any automatic time generator before and when asked about whether they fill this type of product useful for them then 80% of the people responded positively.

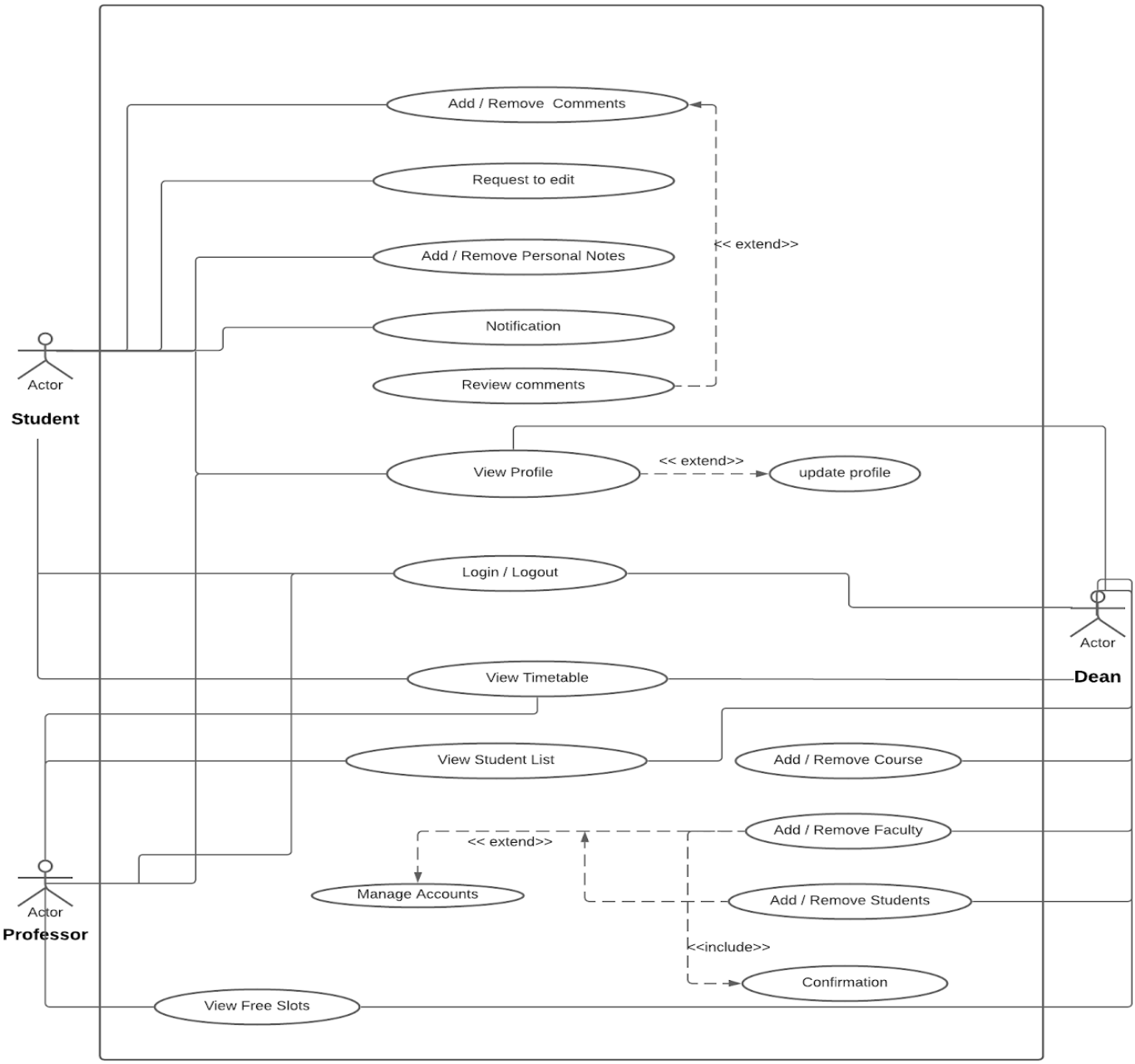


As we all know that most of the universities/organizations provide static time tables and they just take care of clashes in their timetable. Students are averagely satisfied by such a timetable. As mentioned above students need well-distributed loads of lectures and hence need a better algorithm to generate timetables.



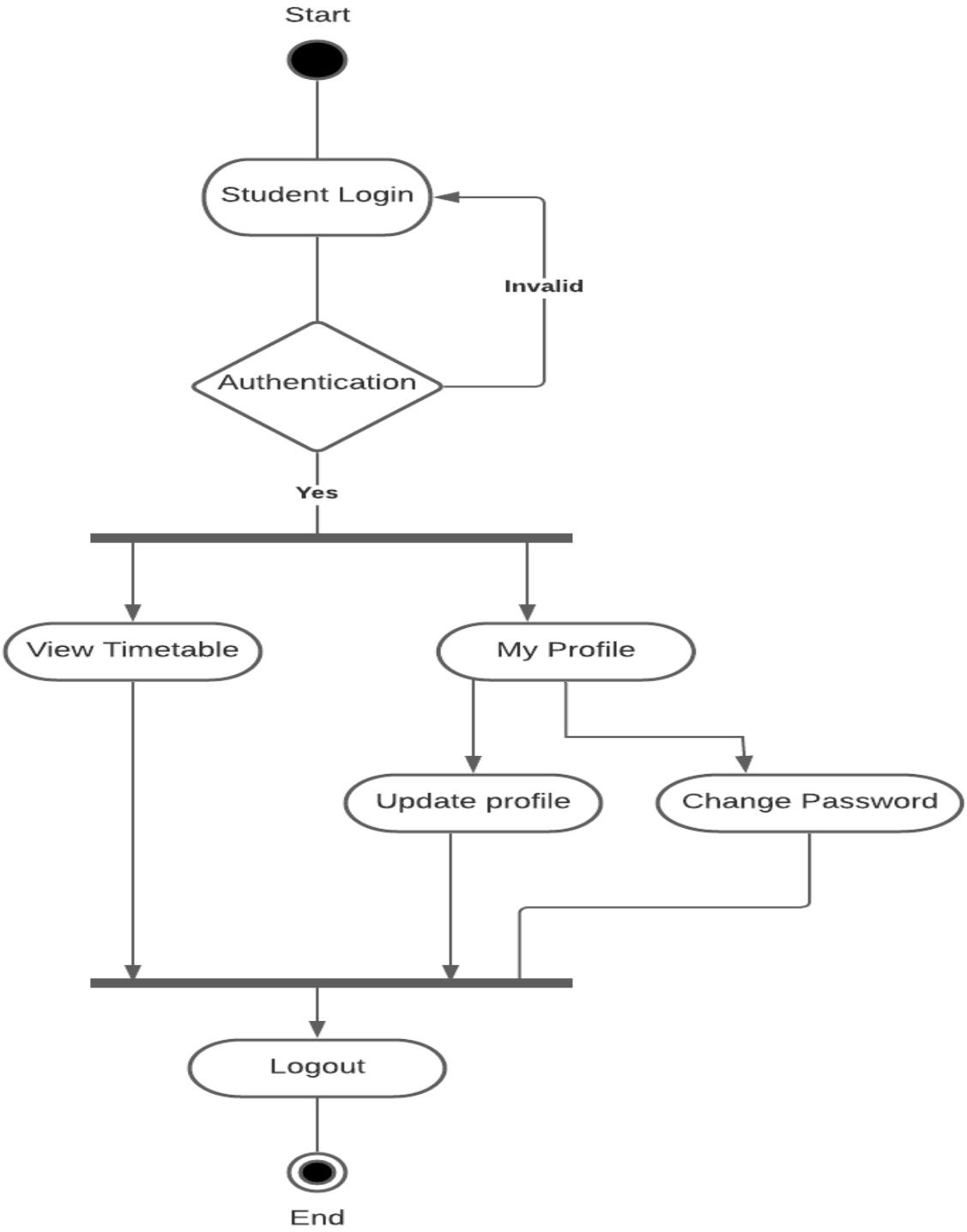
We also wanted to know about the prioritization of requirements. We realized that the most important feature which almost everyone wants is a reminder option for the lecture followed by putting personal notes on a particular lecture. Students also voted for personalized view and dynamicity of timetable in higher amounts.

5) Use-Case Model

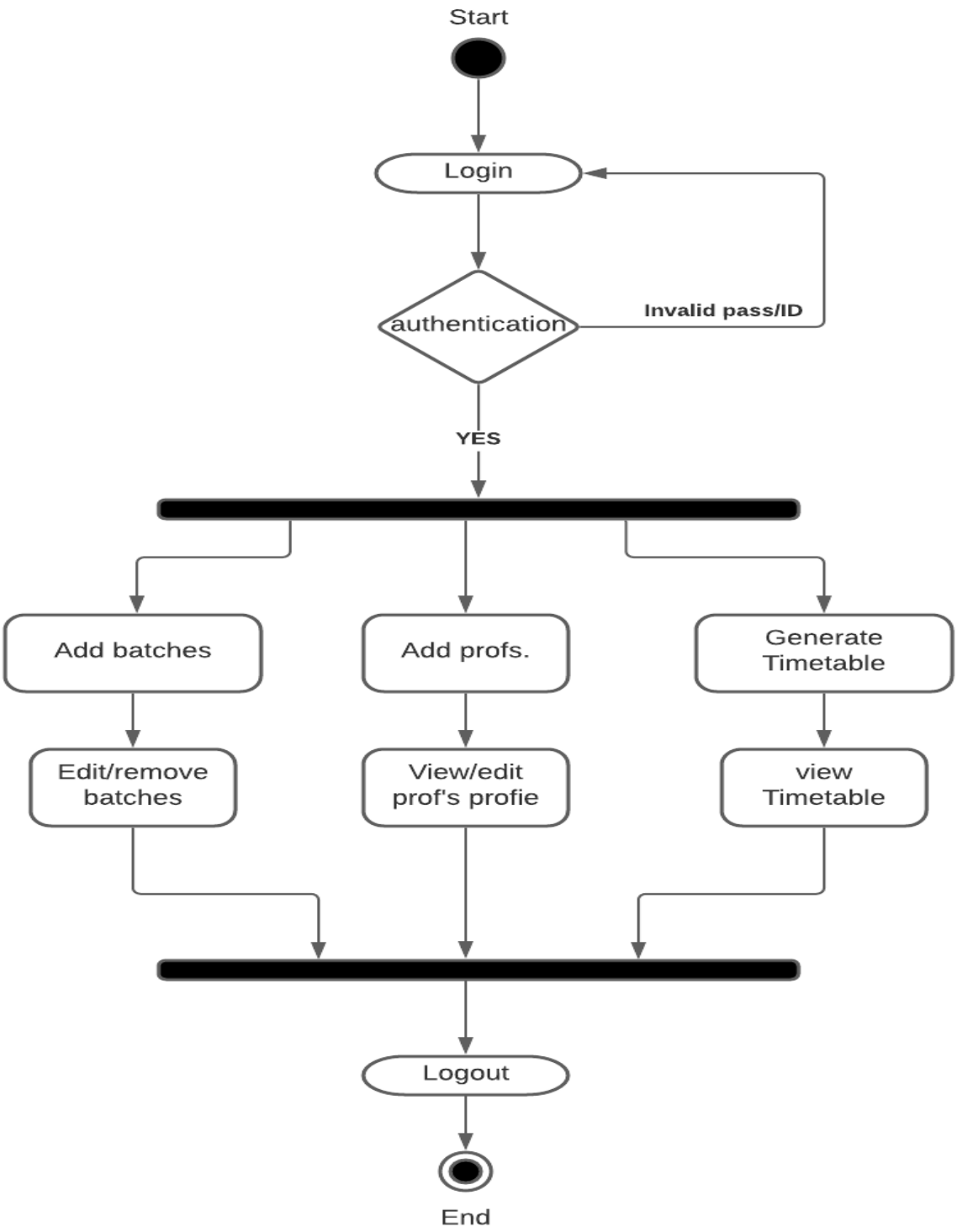


6) Activity Diagrams

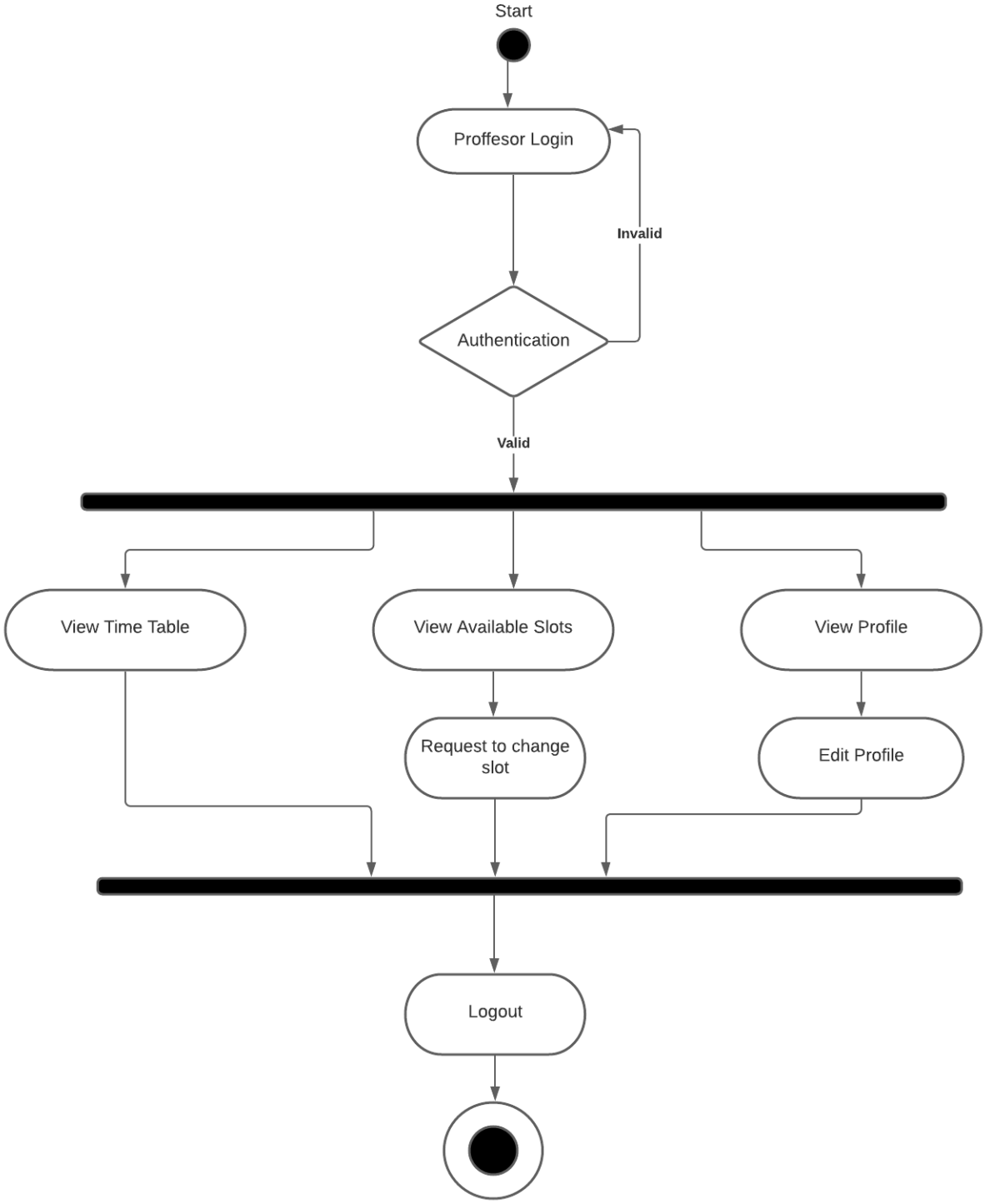
a) Student Activity Diagram



b) Dean Activity Diagram



c) Professor Activity Diagram



7) **Algorithm:**

```
Function main() :
{
        genes(Load CSV File )

        Timetables = generate_random_timetable(population_size, genes)

        While fitness(Timetables) < fitness_threshold & Iterations < max_iterations
        {
                Evolve(Timetables)
                Iterations++
        }
}

Function Evolve ( Timetables) :
{
        Calculate_fitness(Timetables)
        sort_by_fitness(Timetables)

        For number_of_child times :
                Group =  random_select(Timetables,group_size)
                New_timetable = crossover(group.best,group.second_best)
                mutate(New_timetable)
                new_timtables.append(New_timetable)
}

Function Calculate_fitness ( timetable ) :
{
         Fitness_value =   -1* (clashes_for_batches (timetable)*M  +
                           clashes_for_professors(timetable))*M +
                           load_for_batches(timetable) +
                           load_for_students(timetable))

        return Fitness_value
}
```

Function mutate(timetable) :
{

```
if(random_num() <= mutation_rate)
{
    (a,b) = select_two_nonclashing_slots (timetable)
    swap(a,b)
}
```

}

Function crossover(parentA , parentB) :
{

```
(a,b) = select_substring_of_genes()
Gene_from_A = parentA [a:b]
Gene_from_B = parentB [1:a-1] + parentB [b+1:total_genes]
Child =  merge(Gene_from_A,Gene_from_B)

return Child
```

}

8) **Contribution:**

Designing Team:

1) Margi Hingrajia (201801014)
2) Priyal Raj (201801106)
3) Jaykumar Darji (201801460)
4) Pratik Parmar (201801211)

Frontend Team:

1) Maharshi Vaghela (201801216)
2) Manish Khandar (201801222)

Backend Team:

1) Jaydeep Machhi (201801452)
2) Nishant Shah (201801403)
3) Harshil Goti (201801130)
4) Yash Patel (201801134)