Project Analysis Report

CS 487 – Software Engineering

Team B

Robert Judka, Paul Myers, Changyu Wu, Mayank Bansal

1. User Analysis

User Categories

- Undergraduate Students
    - Majority of users
    - Taking 12 – 18 credits
- Graduate Students
    - Only major classes
    - More specialized
- Commuter Students
    - Want to go to school as little days a week as possible
    - As little gaps between classes as possible
- Online Students
    - "Class times" not overlapping with other classes
- "Easy Classes" Students
    - Looking for professors with high RMP (Rate My Professor) scores
    - Don't want to overload hard classes on one day
- Working Students
    - Want either all late classes or all early classes
- "Not Morning" Students
    - as little 8/10 am classes as possible
- Part Time Students
    - Usually commuter students
    - May prefer online classes

User Interactions

- Users will select classes from a list generated from all the available classes from the IIT class database
- Users will be able to order their selected classes with the higher priority classes at the top

- Users will select parameters for optimizing their schedule, and then be able to order those parameters with the higher priority parameters at the top
- Users will be able to view all generated schedules (with most optimized schedules appearing first) and select the schedules they like the most
- Users will be able to download their selected schedules as a PDF file and/or a formatted email to send their advisors

Data Interaction

- IIT class schedules
    - Includes the classes, class times, professors, and location
    - Main data used for the system
    - Users will choose their selected classes
    - Interaction with data occurs with all users
- RMP (Rate My Professor) scores
    - Includes the average score of the professor
    - Data used as optimization parameter for students looking for easier classes
    - Interaction with data will only occur if students choose the specified parameter

2. Functional Requirements

2.1 The system should query IIT's class database and allow students to search and select classes they want to take
2.2 After selecting the classes provided by the IIT's class system, the system should generate optimal schedule(s) based on the parameters students select
2.3 If multiple schedules are available, the system must order the schedules in decreasing level of optimization
2.4 The system must resolve any time conflicts with chosen classes ensuring that two classes aren't selected that start or end within the other's time period
2.5 The students should be able to select options to prioritize classes so that the system first adds high priority classes and then adds the remaining selected classes
2.6 The students should be allowed to choose how many credit hours they are willing to take so the system fills the schedule with classes up to and including that credit limit
2.7 Students should be allowed to save the system generated schedules
2.8 Students should be allowed to generate emails of the generated schedules to be sent to the students' academic advisor for the schedule to be approved

3. Non-functional Requirements

3.1 The System will generate all possible schedules in under 1 minute
3.2 The System will prevent users from doing any invalid operations

3.3 The System will be accessible by anyone
3.4 The System will be easy to use

4. Test Plans

4.1 Functional Requirement 2.1

The system should query IIT's class database and allow students to search and select classes they want to take

- **Scenarios and Use cases**
  Students search for IIT classes with keywords such as class codes (e.g. CS 487) or class titles (e.g. Software Engineering), if class(es) available in database, a list of suggestions will be displayed

- **Usability**
  - Add a placeholder "Search with class title or code" in the text input box to remind students how to search for classes
  - Ignore the cases of students' input and begin searching in the background as soon as students have entered something, once found results, prompt the suggestions for students to choose

- **Exception handling**
  - No internet connection, error message "Failed to connect to internet" should be displayed and no further action should be performed
  - Database access rejected due to maintenance, error message "Database maintenance" should be displayed and no further action should be performed
  - Students search with bad keywords, e.g. a class that IIT doesn't offer, error message "No class found" should be displayed

- **Traceability**
  Create a table with columns "Test case #", "Test Case", "Test Steps", "Test Data", "Expected Result" and "Test Result" to keep track of each test and test each case multiple times, for example:

Test case #	Test Case	Test Steps	Test Data	Expected Result	Test Result
1	No Internet connection	1) Open search page 2) Search	"CS 487" and "Software Engineering"	Display error message "No Internet connection"	Same as "Expected Result"

2	Database access denied	1) Open search page 2) Search	"CS 487" and "Software Engineering"	Display error message "Database maintenance"	Same as "Expected Result"
3	Search with bad keywords	1) Open search page 2) Search	"CS 000" and "Robot"	Display error message "No class found"	Same as "Expected Result"
4	Normal search	1) Open search page 2) Search	"CS 487" and "Software Engineering"	Display suggestion "CS 487: Software Engineering"	Same as "Expected Result"
…	…	…	…	…	…

4.2 Functional Requirement 2.2 & 2.3

After selecting the classes provided by the IIT's class system, the system should generate optimal schedule(s) based on the parameters students select, if multiple schedules are available, the system must order the schedules in decreasing level of optimization

- **Scenarios and Use cases**
  After selecting the classes and choosing optimization parameters, a list of different schedules of the classes will be displayed for students to view

- **Usability**
  - After students press "Finish selecting courses", a menu will be displayed, asking students to choose optimization parameters.
  - Each parameter has a short description below it that briefly explains what it does, e.g. Minimum days — This will generate a schedule that allows you to go to school as little days a week as possible
  - After choosing optimization parameters, an activity indicator and message "Generating schedules with parameters 'Minimum days'" will be displayed if a student chooses "Minimum days" as their optimization parameter
  - Students can cancel anytime during schedule generating

- **Exception handling**
  - Taking too long to generate a schedule due to some unexpected error

- **Traceability**
  Create a table with columns "Test case #", "Test Case", "Test Steps", "Test Data", "Expected Result" and "Test Result" to keep track of each test and test each case multiple times, for example:

A student chooses “CS 485” and “CS 487”
CS 485 schedule: (1) M & W 11:25 – 12:40 (2) T & R 11:25 – 12:40 (3) W 6:25 – 9:05
CS 487 schedule: (1) M & R 10:00 – 11:15 (2) T & F 3:15 – 4:30 (3) T & R 8:35 – 9:50

All possible schedules for these two courses are:

1. CS 485 (1) and CS 487 (1) — 3 days a week, gaps: 10 minutes total
2. CS 485 (1) and CS 487 (2) — 4 days a week, gaps: 0 minutes total
3. CS 485 (1) and CS 487 (3) — 4 days a week, gaps: 0 minutes total
4. CS 485 (2) and CS 487 (1) — 3 days a week, 10 minutes total
5. CS 485 (2) and CS 487 (2) — 3 days a week, 155 minutes total
6. CS 485 (2) and CS 487 (3) — 2 days a week, 190 minutes total
7. CS 485 (3) and CS 487 (1) — 3 days a week, 0 minutes total
8. CS 485 (3) and CS 487 (2) — 3 days a week, 0 minutes total
9. CS 485 (3) and CS 487 (3) — 3 days a week, 0 minutes total

Test case #	Test Case	Test Steps	Test Data	Expected Result	Test Result
1	Minimum days	1) Select courses 2) Select options 3) Generate schedules 4) Check if optimal	“CS 485” and “CS 487” with “Minimum days”	6, 1, 4, 5, 7, 8, 9, 2, 3	Same as “Expected Result”
2	Minimum gaps	1) Select courses 2) Select options 3) Generate schedules 4) Check if optimal	“CS 485” and “CS 487” with “Minimum gaps”	2, 3, 7, 8, 9, 1, 4, 5, 6	Same as “Expected Result”
3	Minimum days & gaps	1) Select courses 2) Select options 3) Generate schedules 4) Check if optimal	“CS 485” and “CS 487” with “Minimum days & gaps”	6, 7, 8, 9, 1, 4, 5, 2, 3	Same as “Expected Result”
4	Minimum gaps & days	1) Select courses 2) Select options 3) Generate schedules 4) Check if optimal	“CS 485” and “CS 487” with “Minimum gaps & days”	7, 8, 9, 2, 3, 1, 4, 5, 6	Same as “Expected Result”
…	…	…	…	…	…

4.3 Functional Requirement 2.4

The system must resolve any time conflicts with chosen classes ensuring that two classes aren't selected that start or end within the other's time period

- **Scenarios and Use cases**
  When selecting a course, if the course does not overlap with any of the chosen courses, then the course will be added to the "selected courses" list

- **Usability**
  - Upon successfully adding a course, a message "course added" will be displayed
  - If all sections of the course overlap with one or more selected courses, an error message "time conflict with course XXX" will be displayed
  - If the error message is displayed, students can choose "This class has high priority over the other" option to replace the other one

- **Exception handling**
  - The class that students want to replace has high priority too, then the newly selected course should not be added to the list

- **Traceability**
  Create a table with columns "Test case #", "Test Case", "Test Steps", "Test Data", "Expected Result" and "Test Result" to keep track of each test and test each case multiple times, for example:

  A student chooses "CS 485", "CS 487" and "CS 440", who have the same schedules
  CS 440 schedule: (1) M & W 11:25 – 12:40 (2) T & R 11:25 – 12:40 (3) W 6:25 – 9:05
  CS 485 schedule: (1) M & W 11:25 – 12:40 (2) T & R 11:25 – 12:40 (3) W 6:25 – 9:05
  CS 487 schedule: (1) M & W 11:25 – 12:40 (2) T & R 11:25 – 12:40 (3) W 6:25 – 9:05

Test case #	Test Case	Test Steps	Test Data	Expected Result	Test Result
1	Add CS 485 with no other classes added	1) Open search page 2) Search with "CS 485" 3) Add course "CS 485"	"CS 485"	"Course added"	Same as "Expected Result"
2	Add CS 487 with CS 485 on the	1) Open search page 2) Search with "CS 485" 3) Add course "CS 485" 4) Search with "CS 487" 5) Add course "CS 487"	"CS 485" and "CS 487"	"Time conflict with CS 485"	Same as "Expected Result"

	selected course list				
3	Give “CS 487” high priority in Test case 2	1) Open search page 2) Search with “CS 485” 3) Add course “CS 485” 4) Search with “CS 487” 5) Add course “CS 487” 6) Choose “high priority”	“CS 485” and “CS 487”	“Course replaced”	Same as “Expected Result”
4	After Case 3, choose “CS 440” and give it high priority	1) Open search page 2) Search with “CS 485” 3) Add course “CS 485” 4) Search with “CS 487” 5) Add course “CS 487” 6) Choose “high priority” 7) Search with “CS 440” 8) Add course “CS 440” 9) Choose “high priority”	“CS 485”, “CS 487” and “CS 440”	“Failed to replace course, CS 487 has high priority”	Same as “Expected Result”
…	…	…	…	…	…

4.4 Functional Requirement 2.5 & 2.6

The students should be allowed to choose how many credit hours they are willing to take so the system fills the schedule with classes up to and including that credit limit. And the students should be able to select options to prioritize classes so that the system first adds high priority classes and then adds the remaining selected classes

- **Scenarios and Use cases**
  Before generating schedules, students may give priorities to each class they have chosen and choose how many credit hours they are willing to take. The generated schedules will have as many high priority classes as possible given that they don’t exceed the credit limit

- **Usability**
  - Students may add colored tags to selected courses. Red means highest priority, blue means high priority and gray means no priority
  - If classes with high priority are not included in a schedule, then alert message “XXX is not included but has high priority”

- **Exception handling**
  - The credit limit is lower than any single course’s credit hour, then error message “credit limit too low” should be displayed

- Students give every class high priority (e.g. Mark each class with red tag), then they will be marked with gray tag automatically and no alert message will be displayed on any schedules
-

- **Traceability**

Create a table with columns “Test case #”, “Test Case”, “Test Steps”, “Test Data”, “Expected Result” and “Test Result” to keep track of each test and test each case multiple times, for example:

A student chooses courses “CS 440”, “CS 485” and “CS 487”
CS 440, credit: 3
CS 485, credit: 4
CS 487, credit: 5

Test case #	Test Case	Test Steps	Test Data	Expected Result	Test Result
1	Credit limit 3, priority: • CS 487 • CS 485 • CS 440	1) Add “CS 440”, “CS 485” and “CS 487” 2) Give them different priorities 3) Press “Generate schedules”	Credit limit 3, priority: • CS 487 • CS 485 • CS 440	See result 1	Same as “Expected Result”
2	Credit limit 6, priority: • CS 487 • CS 485 • CS 440	1) Add “CS 440”, “CS 485” and “CS 487” 2) Give them different priorities 3) Press “Generate schedules”	Credit limit 6, priority: • CS 487 • CS 485 • CS 440	See result 2	Same as “Expected Result”
3	Credit limit 9, priority: • CS 487 • CS 485 • CS 440	1) Add “CS 440”, “CS 485” and “CS 487” 2) Give them different priorities	Credit limit 9, priority: • CS 487 • CS 485 • CS 440	See result 3	Same as “Expected Result”

		3) Press "Generate schedules"			
...	...	...	...	...	...

Result 1:

Schedule 1				
Course Code	Course Title	Credit Hour	Priority	Time
CS 440	Programming Languages	3	●	XXX
Alert: Courses "CS 487" and "CS 485" with high priority are not on the schedule				

Result 2:

Schedule 1				
Course Code	Course Title	Credit Hour	Priority	Time
CS 487	Software Engineering	3	●	XXX
Alert: Course "CS 485" with high priority is not on the schedule				

Schedule 2				
Course Code	Course Title	Credit Hour	Priority	Time
CS 485	Computer and Society	3	●	XXX
Alert: Course "CS 487" with high priority is not on the schedule				

Schedule 3				
Course Code	Course Title	Credit Hour	Priority	Time
CS 440	Programming Languages	3	●	XXX
Alert: Courses "CS 487" and "CS 485" with high priority are not on the schedule				

Result 3:

Schedule 1				
Course Code	Course Title	Credit Hour	Priority	Time
CS 487	Software Engineering	3	●	XXX
CS 485	Computer and Society	3	●	XXX

Schedule 2				
Course Code	Course Title	Credit Hour	Priority	Time
CS 487	Software Engineering	3	●	XXX
CS 440	Programming Languages	3	●	XXX
Alert: Course "CS 485" with high priority is not on the schedule				

Schedule 3				
Course Code	Course Title	Credit Hour	Priority	Time
CS 485	Computer and Society	3	●	XXX
CS 440	Programming Languages	3	●	XXX
Alert: Course "CS 487" with high priority is not on the schedule				

4.5 Functional Requirement 2.7

Students should be allowed to save the system generated schedules

- **Scenarios and Use cases**
  Students review the generated schedules and may choose any one or more to save them locally

- **Usability**
  - Upon pressing “Save schedule”, students are asked to give a name to the schedule, e.g. “Fall 2017”
  - The schedules are saved to local hard drive in “.csv” format, which can be opened by Microsoft Excel and can be imported into Google Calendar

- **Exception handling**
  - Students do not give a name to the schedule when saving it, then a random name “Schedule XXX” will be generated

- **Traceability**
  Create a table with columns “Test case #”, “Test Case”, “Test Steps”, “Test Data”, “Expected Result” and “Test Result” to keep track of each test and test each case multiple times, for example:

Test case #	Test Case	Test Steps	Test Data	Expected Result	Test Result
1	Normal Save	1) Add courses 2) Generate schedules 3) Review schedules 4) Press “Save schedule” 5) Name it “Fall 2017”	“CS 485” and “CS 487”	A “Fall 2017.xlsx” file is saved to local hard drive and contains the schedule generated	Same as “Expected Result”
2	Empty name	1) Add courses 2) Generate schedules 3) Review schedules 4) Press “Save schedule” 5) Press “Complete” without giving it a name	“CS 485” and “CS 487”	A “Schedule XXX.xlsx” file is saved to local hard drive and contains the schedule generated	Same as “Expected Result”

…	…	…	…	…	…

4.6 Functional Requirement 2.8

Students should be allowed to generate emails of the generated schedules to be sent to the students' academic advisor for the schedule to be approved

- **Scenarios and Use cases**
  Students review the generated schedules and may choose any one then press "Email" to email it to their academic advisor for the schedule to be approved

- **Usability**
  - This feature requires students to login (or register if they don't have an account)
  - An input box will pop up, asking students to enter their advisors' email address as well as the message body
  - Upon pressing "send", the email will be sent to the advisor with the student's email that they registered this account with

- **Exception handling**
  - Students try to send emails without having logged in, then error message "Please login first to send emails to others" will be displayed
  - Students enter invalid email address, then error message "Please enter a valid email address" will be displayed

- **Traceability**
  Create a table with columns "Test case #", "Test Case", "Test Steps", "Test Data", "Expected Result" and "Test Result" to keep track of each test and test each case multiple times, for example:

Test case #	Test Case	Test Steps	Test Data	Expected Result	Test Result
1	Email without having logged in	1) Select courses and generate schedules 2) Pick a schedule and press "email"	"CS 487"	"Please login first to send emails to others"	Same as "Expected Result"
2	Invalid email address	1) Select courses and generate schedules 2) Pick a schedule and press "email"	"CS 487" and cs487@iit.edu, which is an	"Please enter a valid	Same as "Expected Result"

		3) Enter an invalid email address	invalid email address	email address"	
…	…	…	…	…	…

4.7 Non-functional Requirement 2.1

The System will generate all possible schedules in under 1 minute

- **Scenarios and Use cases**
  Students press "Generate schedules" when they've finished selecting courses, then schedules must be generated under 1 minute

- **Usability**
  - There is a cancel button on the screen when generating schedules, allowing students to cancel anytime they want if they do not wish to wait anymore, or they wish to change the selected courses

- **Exception handling**
  - If taking longer than 1 minute to generate a course, then display error message "This is taking too long, try to eliminate some optimization parameters and try again"

- **Traceability**
  Add different courses and choose different optimization parameters to generate schedules many times, document the time taken

4.8 Non-functional Requirement 2.2

The System will prevent users from doing any invalid operations

- **Scenarios and Use cases**
  Students may perform some invalid operations when using the system, for example, adding a course more than once, operations like this will be ignored

- **Usability**
  - Display an error message whenever an invalid operation is performed, reminding students that the operation cannot be done, however, the error message will not block the entire page, but leave students free to perform other operations

- **Exception handling**
  - All invalid operations will cause an error message to display

- **Traceability**
  Ask several students to use the system “abnormally”, for example, giving each selected course a high priority, and document each of the actions and outcome

4.9 Non-functional Requirement 2.3

The System will be accessible by anyone

- **Scenarios and Use cases**
  Everyone can get access to the system. They can search for courses, add courses, choose optimization parameters, and generate schedules without any problem

- **Usability**
  - For the most part, this system does not require users to have an account unless they want to email it to their advisor for approval
  -
- **Exception handling**
  - Database access denied due to maintenance, then the system cannot be used by anyone, error message “database maintenance, please try again later” will be displayed
  - Users try to send emails without having logged in