ITLC Timetable system

Luke Halpenny

Joshua Atunuvbare

Rodions Baranikovs

Contents

[Project Process 3](#_Toc5866460)

[Analysis of Existing System 4](#_Toc5866461)

[Interviews 7](#_Toc5866462)

[Personas 8](#_Toc5866463)

[Requirements 10](#_Toc5866464)

[Use Case Analysis 11](#_Toc5866465)

[Class Diagram 19](#_Toc5866466)

[Sequence Diagrams 20](#_Toc5866467)

[Testing 21](#_Toc5866468)

[Project Log 23](#_Toc5866469)

# Project Process

The software development process that we have chosen for our project is the Waterfall Model.

Advantages of Waterfall Model

We have chosen this model as the requirements are well understood and unlikely to change radically during development, problems in the projects are more likely to be sorted out early on and the model is rigid; making it easier to handle.

VS Iterative Development

Iterative development did not suit our project because there is a risk that with this process, a significant amount of resources could be wasted in making changes to initial design whereas the waterfall model asks for specific deliverables at each stage of process, making sure that each part of the project is fully complete.

VS Incremental Development

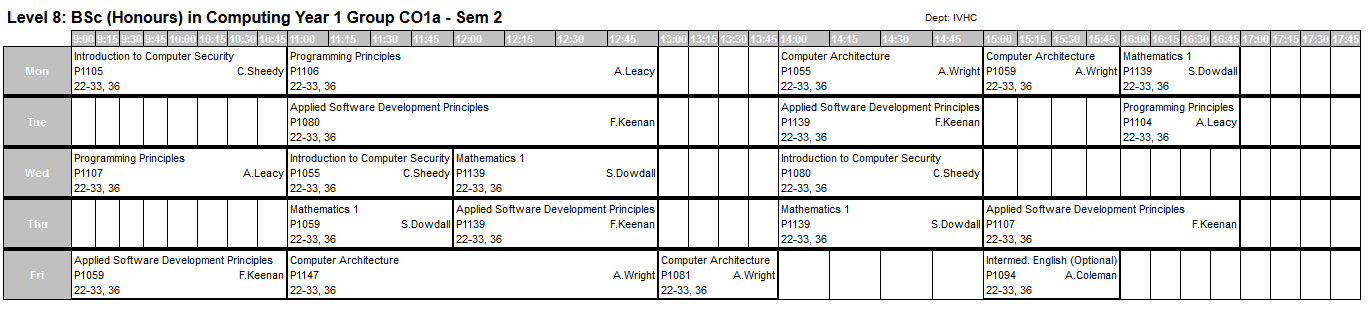
Incremental Development did not suit our project because it is not as cost-efficient as the waterfall model due to frequent software changes that would be avoided in the waterfall model and the process is not visible, making it harder to keep track of progress of the project.

Requirements Elicitation Techniques:

We used multiple techniques to create our list of requirements for the project, including analysing an existing system and interviewing someone who would be using the project.

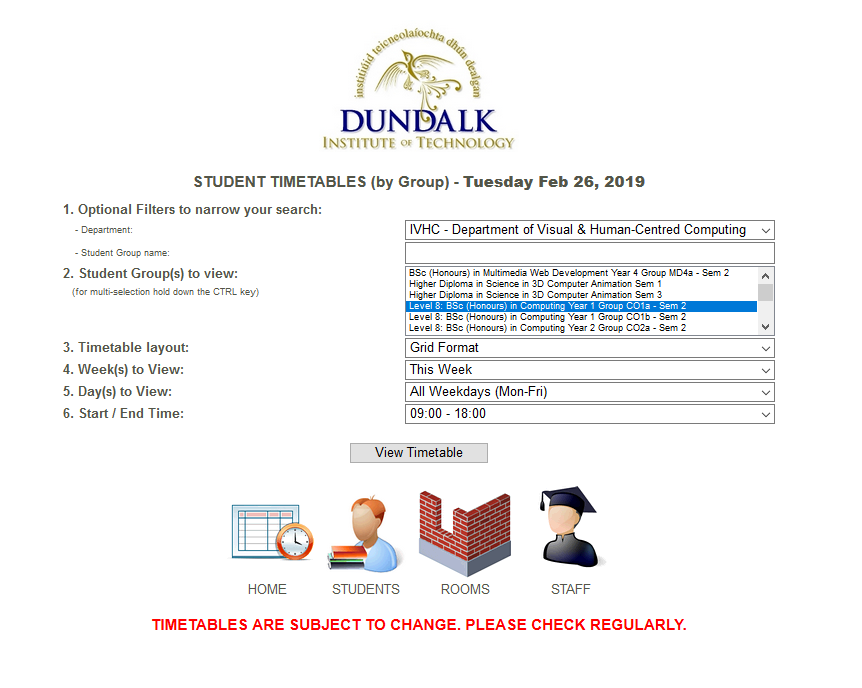
# Analysis of Existing System

Example timetable:



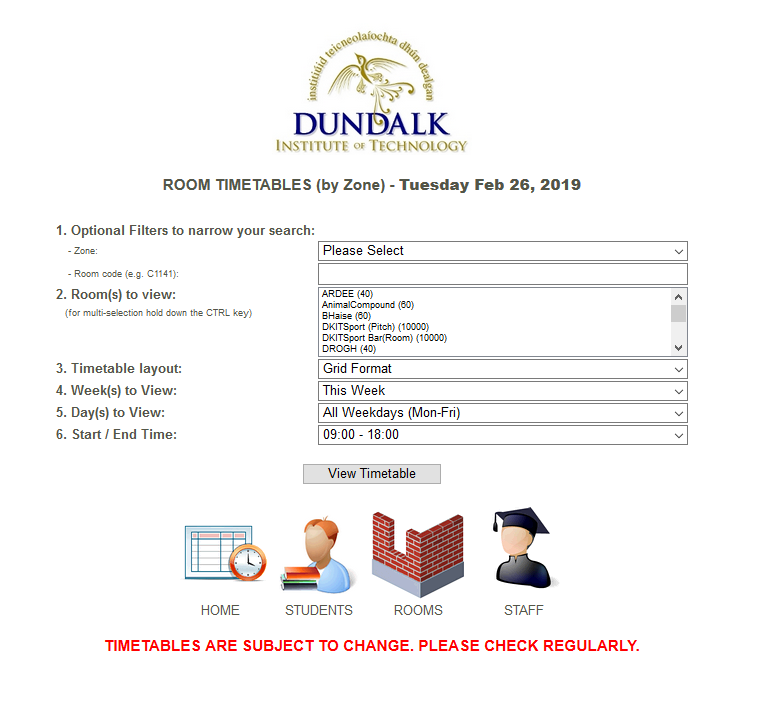
* Display timetable

Student timetable menu:



* Search timetables by student group
* Select display type (grid/list)
* Select time/date

Room timetable menu:



* Search timetables by room
* Select display type (grid/list)
* Select time/date

ITLC Timetable (sent by weekly email):

# Interviews

Mentor

1. How do you use the current DkIT timetable system?

Check classes, check room availability

1. What do you like about it?

Flexibility, Ease of use

1. What could be improved?

Max of three timetables at a time on DkIT system, hard to organise multiple classes as an ITLC mentor

1. What features would you like to see in a timetable system for the ITLC?

Interface with DkIT / cross reference, booking system (options, times, e.g.), live updates, easily seeing currently booked classes

1. How would you use it?

Book classes during free time

Other notes:

Some students are reluctant to email to book classes, having a web-based booking system would be greatly beneficial.

Student

1. What do you like about our current timetable system?

Ease of use, flexible interface (layout preferences) and reliability

1. What do you not like about our current timetable system?

Main page doesn't 'remember' search filters (department, group, layout)

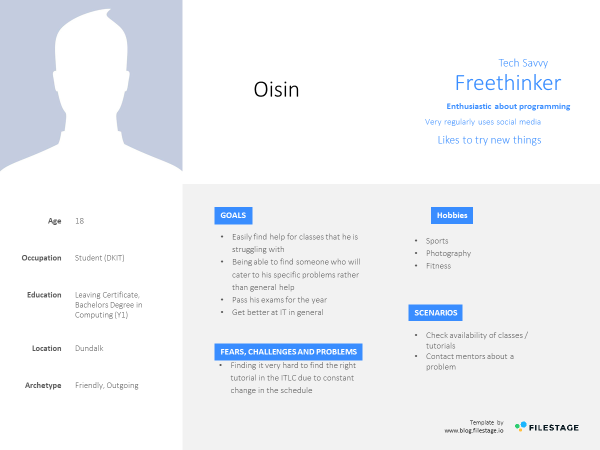
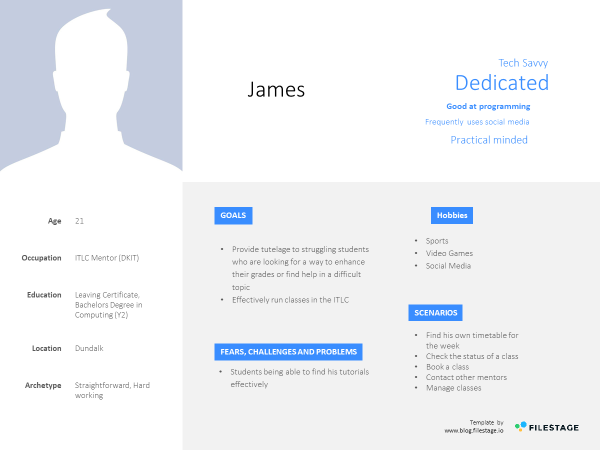
1. What would you add to the current timetable system?

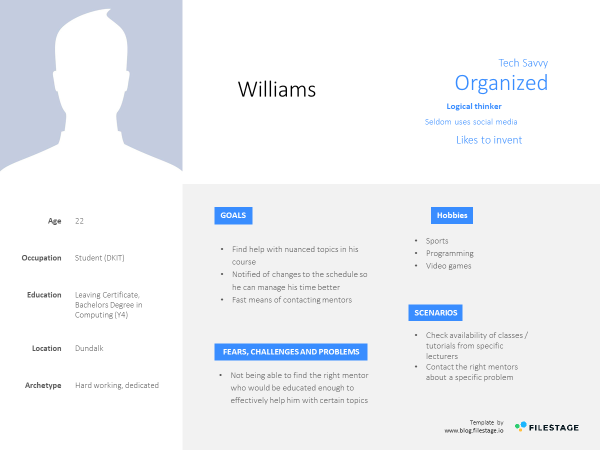
Email or text message notifications regarding changes in the timetable (class cancellation/addition etc)

1. What would you like to improve in the ITLC timetable system?

Message mentors, book or cancel tutorials, notify about changes/ booked tutorials/events

# Personas





# Requirements

User groups

* First Year students
* Final Year students
* Mentors at the ITLC

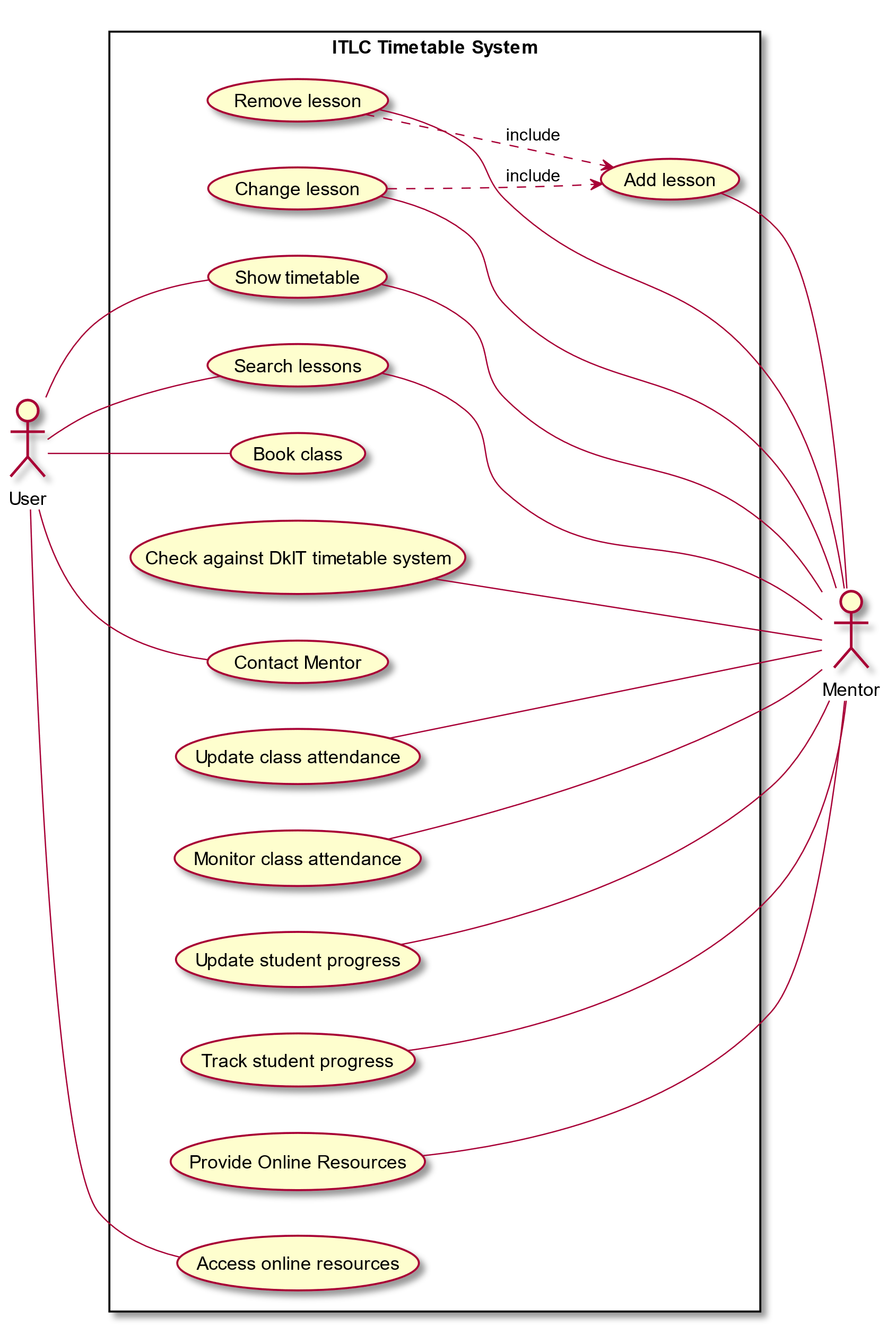
Functional requirements

* Show timetable
* Add lesson
* Remove lesson
* Change lesson
* Search lessons
* Request lesson
* Book classes
* Check against DkIT timetable system
* Check currently booked classes
* Ability to make direct contact to mentors in ITLC
* Notify users
* Monitor class attendance
* Track Progress
* Provide Online Resources

Non-functional requirements

* Security
* Performance (Response time, Ease of use etc.)
* Reliability
* Maintainability

# Use Case Analysis



Add Lesson

Main Actor: Mentor

Preconditions:

- Signed in as mentor

Normal Sequence of Steps:

1. Get lesson info (topic, time, mentor(s), room, recommended level(1st year, 4th year, etc.))

2. Add lesson to schedule

Postconditions:

- Updated timetable

- Time-slot cannot be booked again

Remove Lesson

Main Actor: Mentor

Preconditions:

- Signed in as mentor

Normal Sequence of Steps:

1. Select lesson to Remove

2. Update schedule

3. Inform involved users (mentor, students who booked)

Postconditions:

- Updated timetable

- Time-slot can be booked again

Change Lesson

Main Actor: Mentor

Preconditions:

- Signed in as mentor

Normal Sequence of Steps:

1. Select lesson to change

2. Get new information (new time, changed room, etc.)

3. Update schedule

4. Inform involved users (mentor, students who booked)

Postconditions:

- Updated timetable

- New time-slot cannot be booked again

- Old time-slot can be booked again

Show Timetable

Main Actor: User, Mentor

Preconditions:

Normal Sequence of Steps:

1. Select timetable to show

2. Display timetable

Postconditions:

Search Lessons

Main Actor: User

Preconditions:

Normal Sequence of Steps:

1. Get search terms (lesson type, time, etc.)

2. Search schedule for matching classes

3. Display matching classes

4. Student may show interest in attending, add to list

Postconditions:

Book Class

Main Actor: User

Preconditions:

Normal Sequence of Steps:

1. Get information (topic, preferred time, etc.)

2. If class already exists, add to list of students attending

3. Otherwise, add to request list

3. Mentor contacts student if additional info needed (e.g. different time)

4. Class added to schedule

Postconditions:

- New class booked

- Time-slot cannot be booked again

Check against DkIT Timetable

Main Actor: Mentor

Preconditions:

Normal Sequence of Steps:

1. Select DkIT classes/rooms to search

2. Display common free slots

3. Display links to individual class/room timetables

Postconditions:

|  |  |
| --- | --- |
| **Title:** | Contact Mentor |
| **Description:** | Allow students to contact mentors to ask follow-up questions from lectures |
| **Primary Actor:** | User |
| **Pre-conditions:** | User has booked class with mentor. |
| **Post-conditions:** | Message has been sent to mentor. Notification has been sent to mentor. |
| **Sequence of events:** | User finds mentor on system.  User sends message via system. |
| **Extensions:** | Message fails to send to mentor. |

|  |  |
| --- | --- |
| **Title:** | Update class attendance |
| **Description:** | System updates number of class participants as user signs up |
| **Actors:** | System, User |
| **Pre-conditions:** | Class has been made. |
| **Post-conditions:** | Updated class attendance |
| **Sequence of events:** | User books class on system System updates number of users attended class |

|  |  |
| --- | --- |
| **Title:** | Monitor class attendance |
| **Description:** | Allows mentor to check current attendance of class. |
| **Primary Actor:** | Mentor |
| **Pre-conditions:** | Class has been made |
| **Post-conditions:** |  |
| **Sequence of events:** | Mentor uses system to monitor class attendance |

|  |  |
| --- | --- |
| **Title:** | Update student progress |
| **Description:** | Shows information on classes attended by the students / grades of the student |
| **Actors:** | User, System |
| **Pre-conditions:** | Student has attended at least one class  Students consents to progress tracking |
| **Post-conditions:** | Updated student progress |
| **Sequence of events:** | Select student and class  Get student progress (grades etc.)  Update database |

|  |  |
| --- | --- |
| **Title:** | Track class attendance |
| **Description:** | Track information on student progress |
| **Actors:** | System, User |
| **Pre-conditions:** | Student has attended at least one class  Students consents to progress tracking |
| **Post-conditions:** |  |
| **Sequence of events:** | Student progress is updated System compares user latest statistics to that of previous progress updates System makes this comparison available to user and relevant mentors. |

Provide Online Resources

Preconditions:

- Signed in as mentor

- Have resource to provide

Normal Sequence of Steps:

1. Select topic

2. Get resource (text, link, pdf, etc.)

3. Add to resource list for that topic

4. Edit and save the topic

Postconditions:

- Resource is up online

- Sign out

Access Online Resources

Preconditions:

- Signed in as a student

Normal Sequence of Steps:

1. Get topic to search for

2. Display resources for selected topic

3. download/save the resources

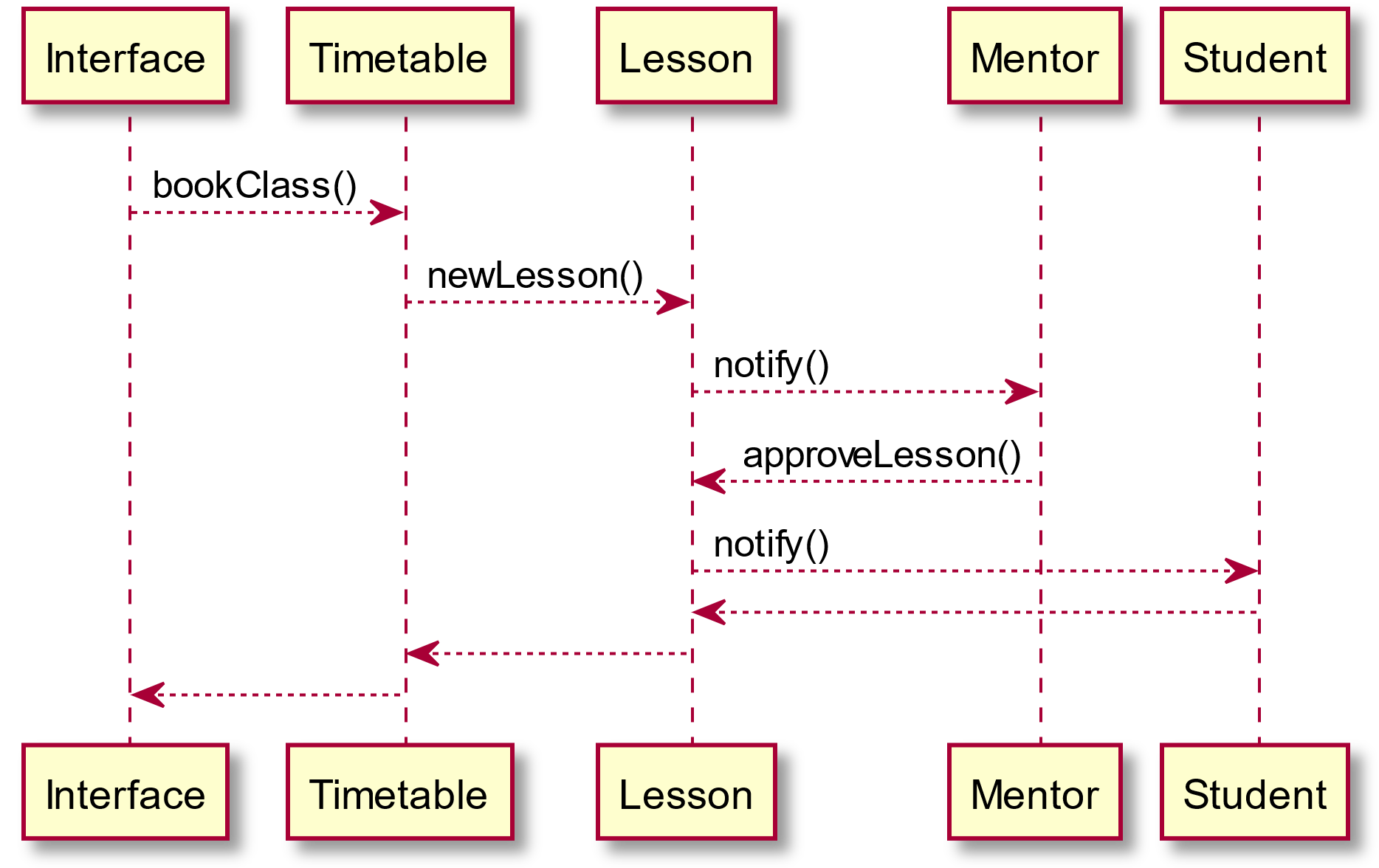
Postconditions:

Resource obtained

# Class Diagram

# Sequence Diagrams

bookClass():



# Testing

Timetable Object:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| id | Date | Time | Length | Topic | Room | Mentor | Level |
| 0 | 25/03/2019 | 11:00 | 60 | Java | P1104 | Anne Leacy | 1st Year |
| 1 | 25/03/2019 | 09:00 | 60 | OS Development | P1106 | Dave Strider | 3rd Year |
| 2 | 25/03/2019 | 11:00 | 120 | Web Programming | P1080 | Derek O'Reilly | 2nd Year |
| 3 | 26/03/2019 | 14:00 | 30 | C++ | P1147 | John Egbert | 3rd Year |
| 4 | 26/03/2019 | 11:00 | 60 | Java | P1165 | Anne Leacy | 2nd Year |
| 5 | 27/03/2019 | 11:00 | 120 | Computer Architecture | P1081 | Andrew Wright | 1st Year |
| 6 | 27/03/2019 | 13:00 | 90 | Compilers | P1105 | Rose Lalonde | 4th Year |
| 7 | 27/03/2019 | 14:00 | 60 | Web Development | P1106 | Jade Harley | 2nd Year |

anyMatchingRoom(String room):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case | Room | Expected Result | Actual Result | Pass/Fail |
| 1 | P1104 | TRUE | TRUE | Pass |
| 2 | P1105 | TRUE | TRUE | Pass |
| 3 | P1106 | TRUE | TRUE | Pass |
| 4 | P1107 | FALSE | FALSE | Pass |
| 5 | P1080 | TRUE | TRUE | Pass |
| 6 | P1081 | TRUE | TRUE | Pass |
| 7 | F1234 | FALSE | FALSE | Pass |
| 8 | P1165 | TRUE | TRUE | Pass |
| 9 | P1145 | FALSE | FALSE | Pass |
| 10 | P1147 | TRUE | TRUE | Pass |
| 11 | ROOM | FALSE | FALSE | Pass |
| 12 | abcde | FALSE | FALSE | Pass |

allocatePlaces(Year, Topic):

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Year | Topic | Expected Result |
| 1 | 1 | Java | 20 |
| 2 | 2 | Java | 15 |
| 3 | 3 | Java | 10 |
| 4 | 4 | Java | 5 |
| 5 | 1 | Maths | 30 |
| 6 | 2 | Maths | 22 |
| 7 | 3 | Maths | 15 |
| 8 | 4 | Maths | 8 |
| 9 | 1 | Web | 16 |
| 10 | 2 | Web | 12 |
| 11 | 3 | Web | 8 |
| 12 | 4 | Web | 4 |
| 13 | 1 | C++ | 20 |
| 14 | 2 | C++ | 15 |
| 15 | 3 | C++ | 10 |
| 16 | 4 | C++ | 5 |

# Project Log

|  |  |  |
| --- | --- | --- |
| Section | Name | Date |
| Presentation | Luke Halpenny Joshua Atunuvbare  Rodions Baranikovs | Feb 21 |
| Analysis: Existing System | Luke Halpenny | Feb 26 |
| Requirements List | Luke Halpenny Joshua Atunuvbare  Rodions Baranikovs | Feb 26 – Mar 6 |
| Personas + Interviews | Luke Halpenny Joshua Atunuvbare  Rodions Baranikovs | Feb 28 Mar 6 Mar 5 |
| Project Process | Joshua Atunuvbare | Mar 6 |
| Use Cases | Luke Halpenny Joshua Atunuvbare  Rodions Baranikovs | Mar 7 – Mar 15 |
| Class Diagram | Luke Halpenny Joshua Atunuvbare  Rodions Baranikovs | Mar 21 – Mar 26 |
| Sequence Diagram | Luke Halpenny | Apr 11 |
| Implementation | Luke Halpenny Joshua Atunuvbare  Rodions Baranikovs | Mar 28 – Apr 8 |
| Testing | Luke Halpenny | Apr 2 – Apr 11 |