**School Management System (SMS)**

**App**

* **Module 2 – Sections and Schedule Management**



**CMPS 350 Project Phase 1 – WebApp UI Design and Implementation**

|  |  |
| --- | --- |
| **Group Id:** | G4 |
| **Group Members:** | Somaya  Muraam  Duaa |

**Grading Rubric - In the Functionality column please specify either: *Working (completed x%)*, *Not Working (completed x%)* or *Not done*.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Criteria | % | Functionality\* | Quality of the implementation | Grade |
| 1) Design the app Web UI and navigation (10%).  2) Implement the app Web UI and navigation using HTML, CSS and JavaScript (30%)  3) Implement the client-side data access repositories to read/write the app data from/to IndexedDB (40%) | | **80%** | Done and working | - For get sections, only show current year sections by default,  - For adding courses to a section, it is better to select a subject once then set to days and times the class will be offered (need to support offering a subjects multiple times per week for a section).  - Copy from previous year, placement of the button and hiding the button behave can be improved.  - For the teacher, by default show the current year schedule.  - See further comments added to the screenshots in the testing section. | 77 |
|  | Login | 5 | Done and working | let teacher = await getTeacher(user.userName)  let parent = await getParent(user.userName)  let principal = await getPrincipal(user.userName)  Above is NOT efficient, try teacher if fails, then parent otherwise principal. No need to try all the 3 of them. |  |
|  | Get sections (+ providing links to navigate to other functions) | 10 | Done and working | getSections poorly done. You should NOT read all then filter. Must read from DB only sections matching year and grade. |  |
|  | Add section | 8 | Done and working |  |  |
|  | Update section | 5 | Done and working |  |  |
|  | Delete section | 5 | Done and working |  |  |
|  | View section | 5 | Done and working |  |  |
|  | Section students (List, Add, Delete) | 15 | Done and working |  |  |
|  | Section courses (List, Add, Delete) | 15 | Done and working | - Subjects, years, grades, days, fromTimes, endtimes should NOT be hardcoded in the app.  - fromTimes, endTimes could be merges into sessions (e.g., 8am to 10am, 10:15am to 12pm). |  |
|  | Copy sections setup from last year | 9 | Done and working |  |  |
|  | Get timetable (Principal, Teacher, Parent) | 15 | Done and working | Scary and complicated. I am sure it could be simplified. |  |
|  | Overall App design and navigation | 8 | Done and working | Excellent.  Not sure why every page call getDB()!!! Do not see the need for it. |  |
| **4) Application Design:** Entities Class Diagram and the Repositories Class diagram. | | **8** | Done and working | Excellent work but few issues identified. See the comments added to the diagrams. | 7 |
| 5) Create test data JSON files for your module entities | | 6 | Done and working | Excellent | 6 |
| **6) Testing documentation** using screen shots illustrating the testing results.  - Discussion of the project contribution of each team member. | | 6 | Done and working | Excellent | 6 |
| **Total** | | 100 |  | Excellent work | 96 |

**\* Possible grading for functionality** - ***Working*** (get 60% of the assigned grade), ***Not*** ***working*** (lose 40% of assigned grade and ***Not done*** (get 0). The remaining grade is assigned to the quality of the implementation.

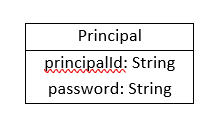
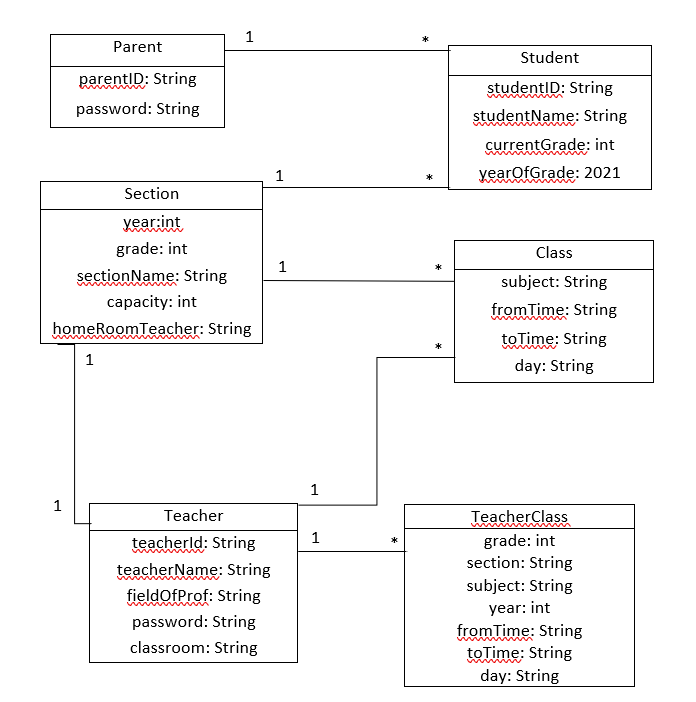
In case your implementation is not working then 40% of the grade will be lost and the remaining 60% will be determined based on of the code quality and how close your solution to the working implementation.

Solution quality also includes meaningful naming of identifiers (according to Android naming conventions), no redundant code, simple and efficient design, clean implementation without unnecessary files/code, use of comments where necessary, proper code formatting and indentation.

**Marks will be reduced** for **code duplication**, poor/inefficient coding practices, poor naming of identifiers, unclean/untidy submission, and unnecessary complex/poor user interface design.

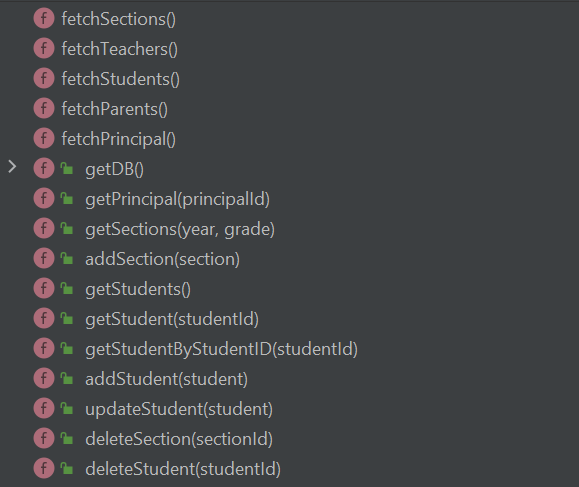
# Application Design

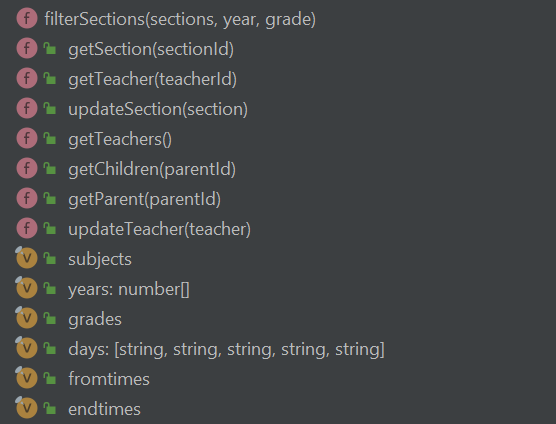
# Entities Class Diagram



# Repositories Class Diagram

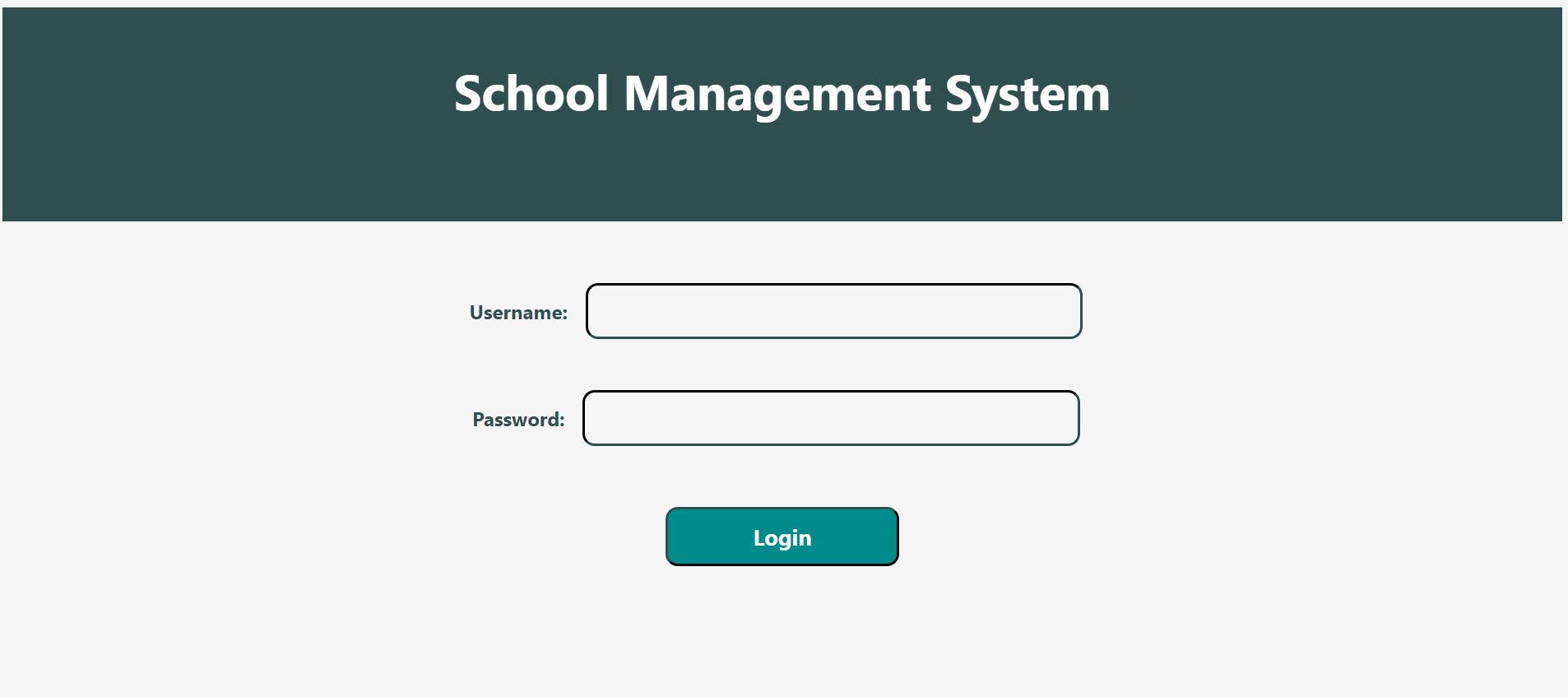
Repository.js:



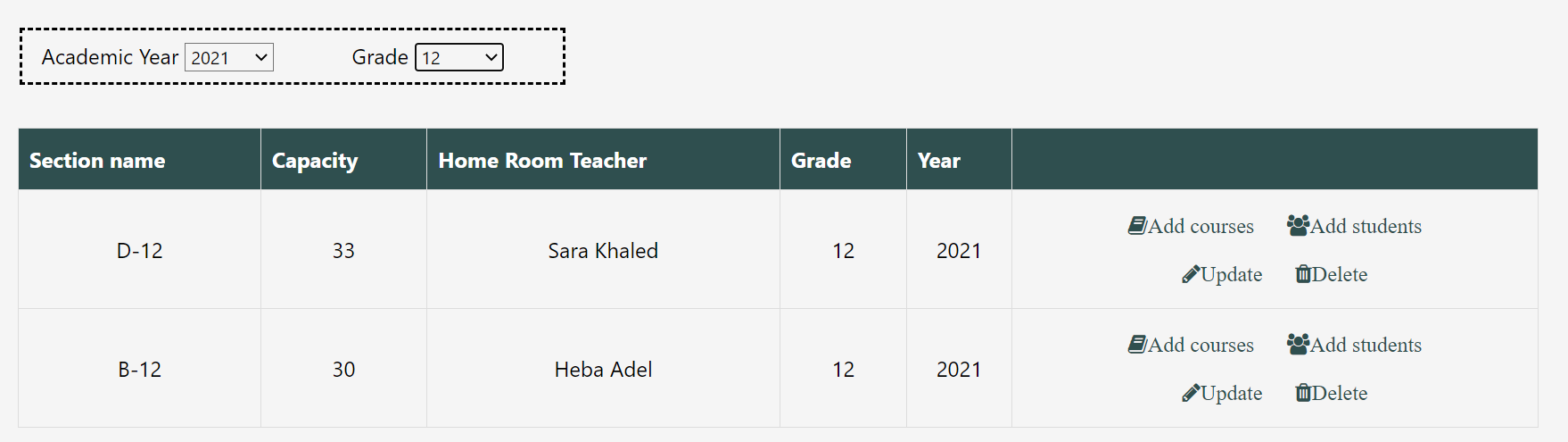


# App Testing

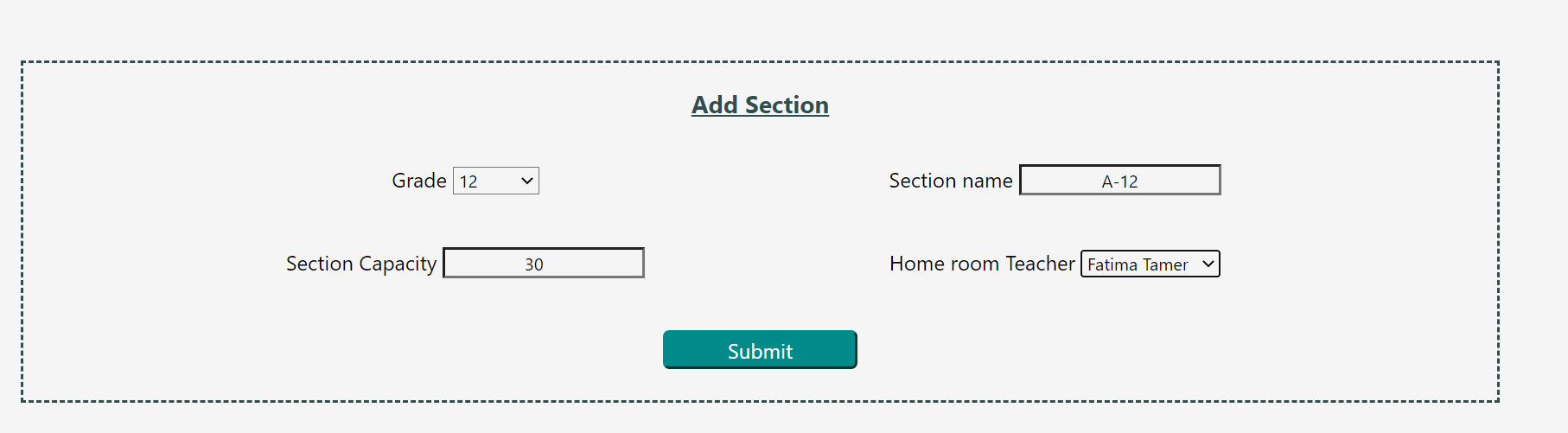
# Login



# Get sections (+ providing links to navigate to other functions)

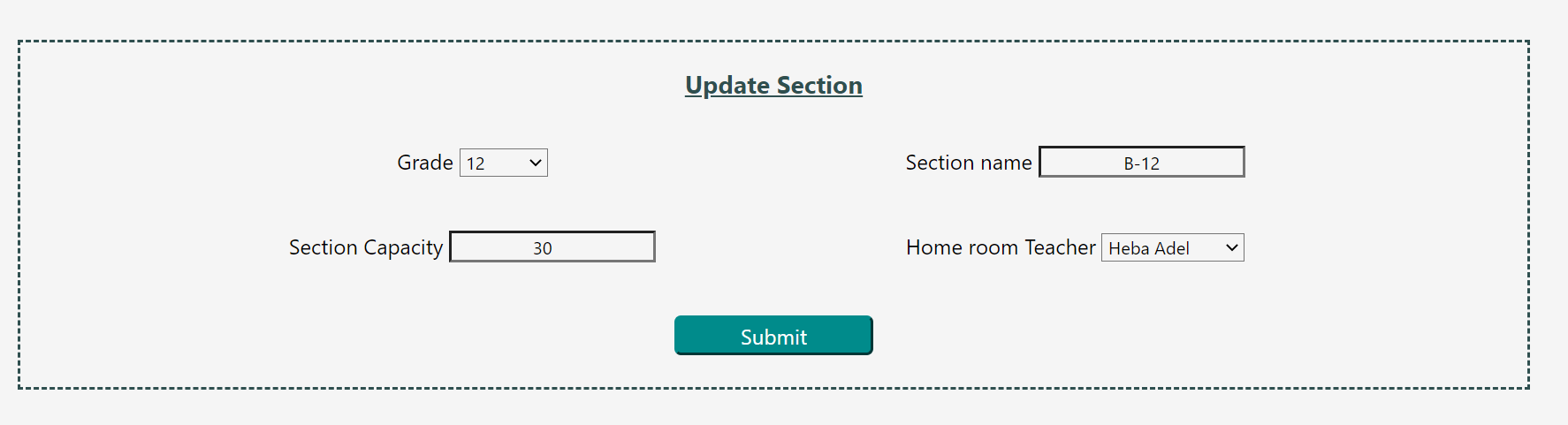


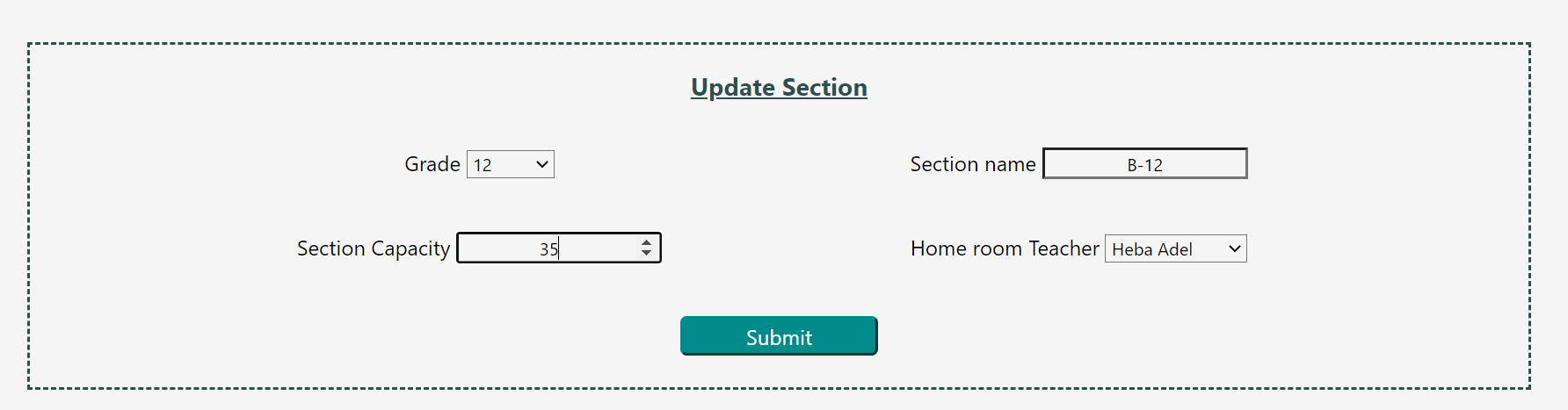
# Add section





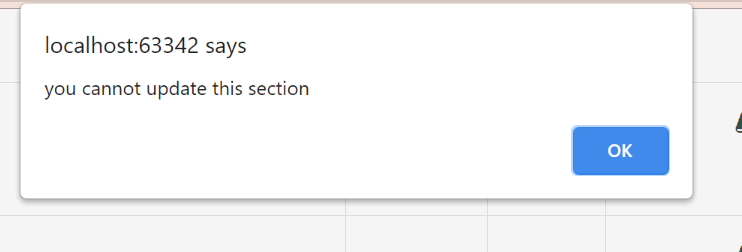
# Update section



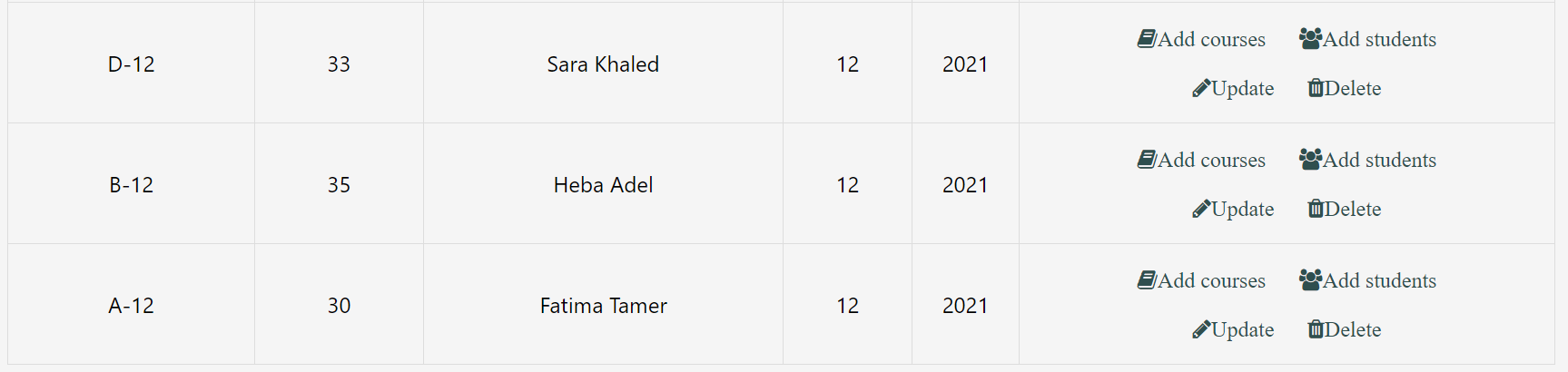


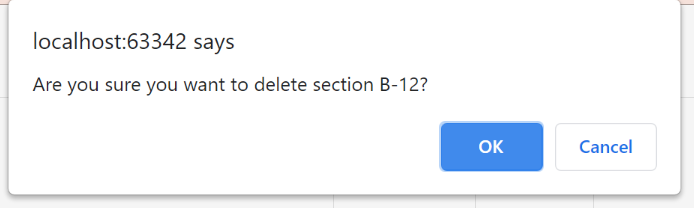


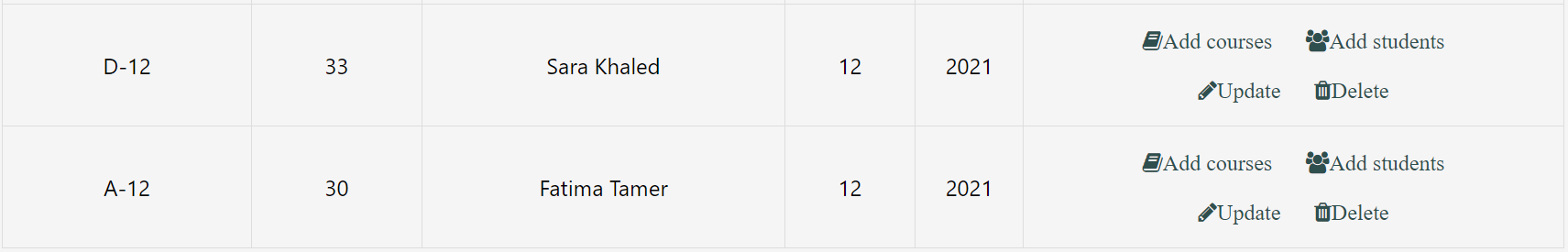
If the principal wants to update an old section, it will give him alert message that he cannot update.



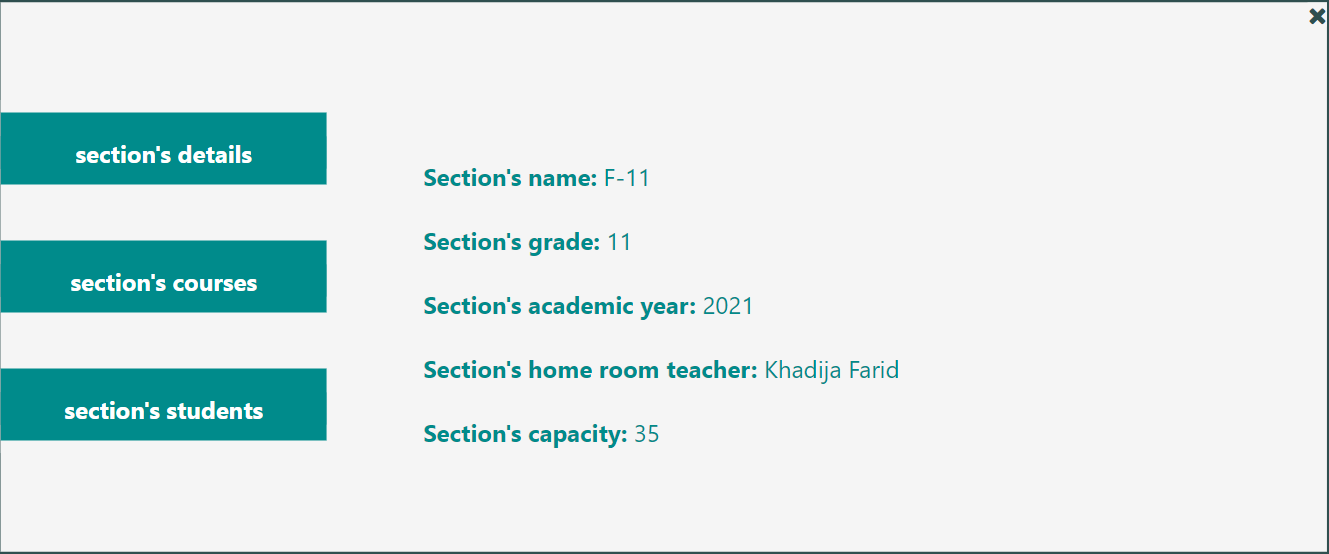
# Delete section



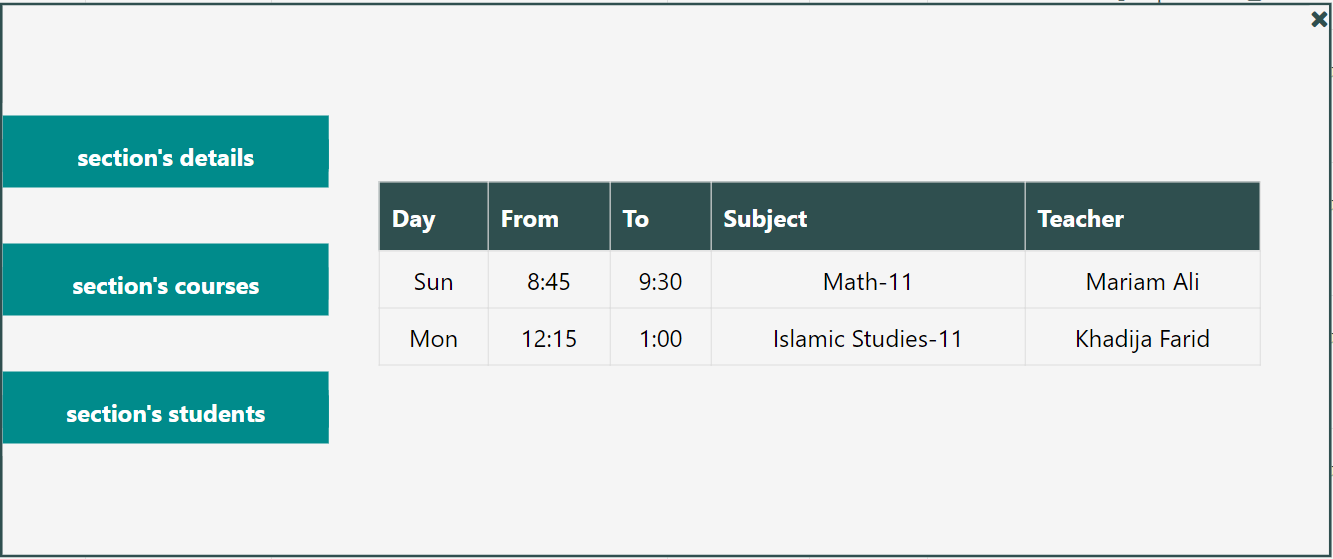


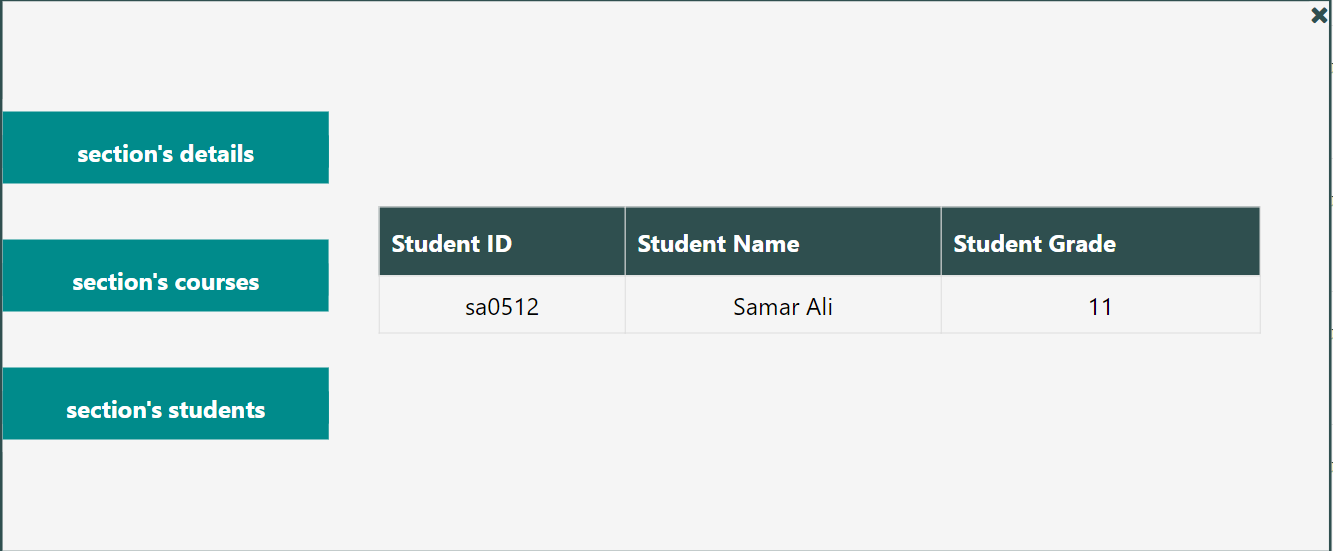


# View section



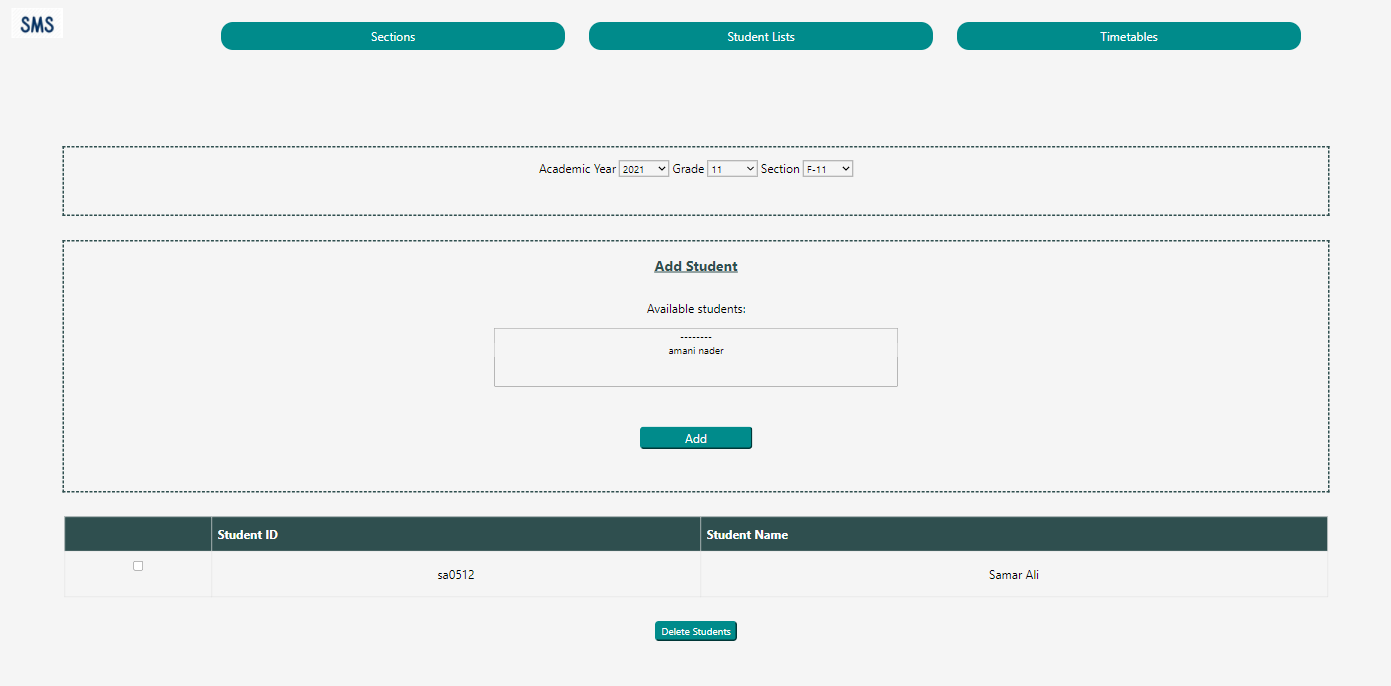




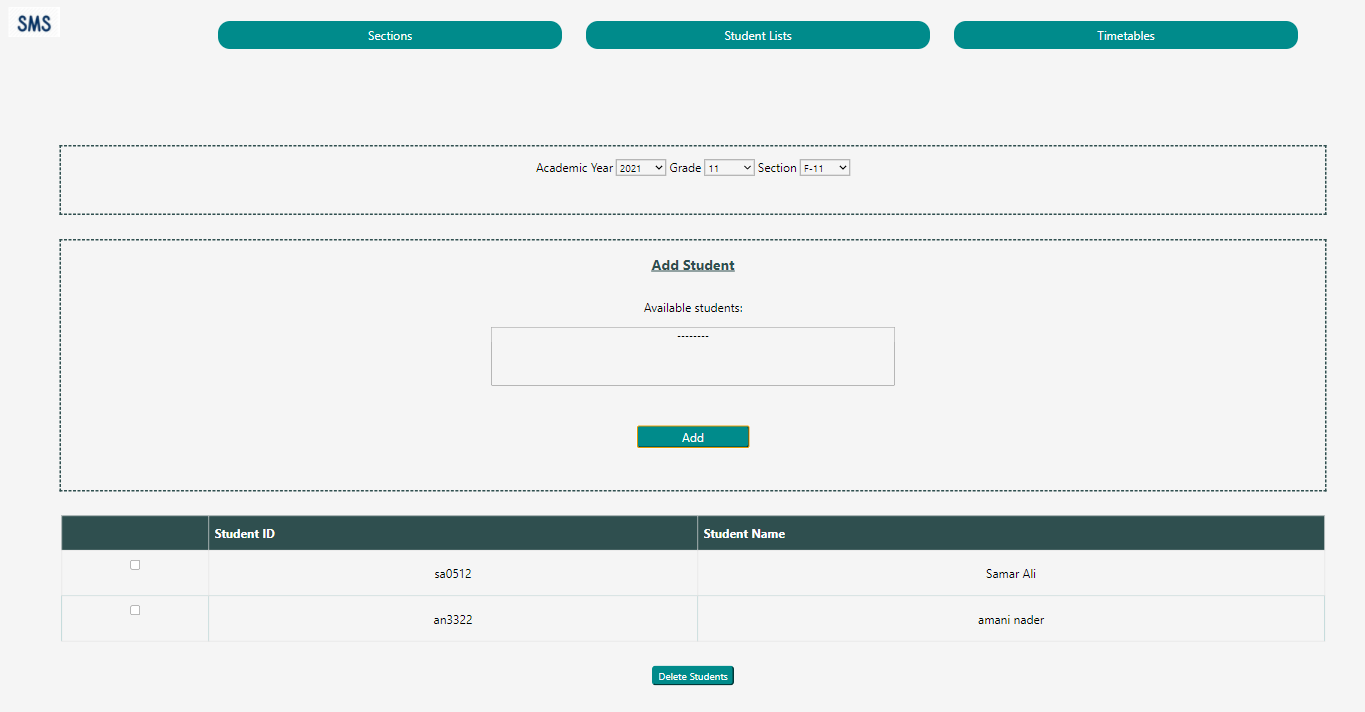


# Section students (List, Add, Delete)

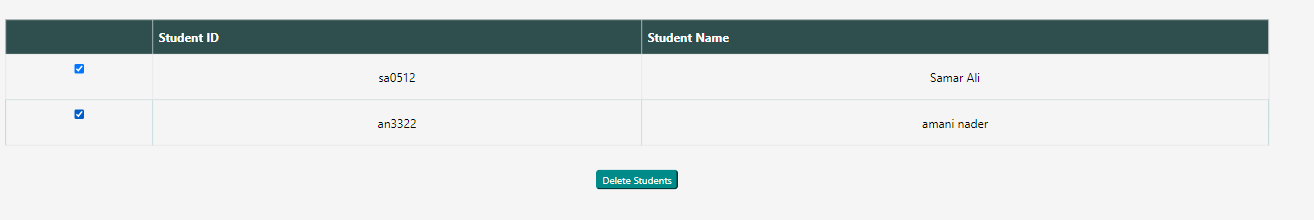
List:

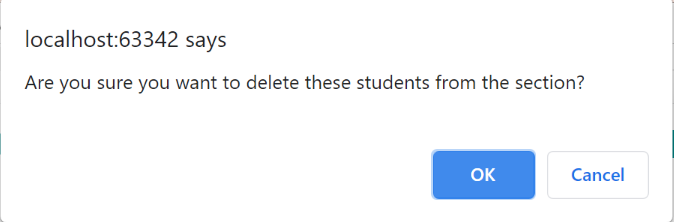


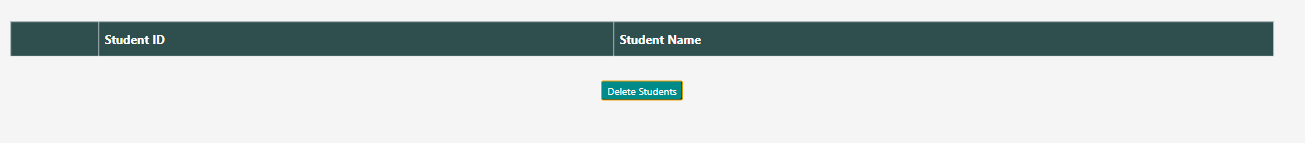
Add:



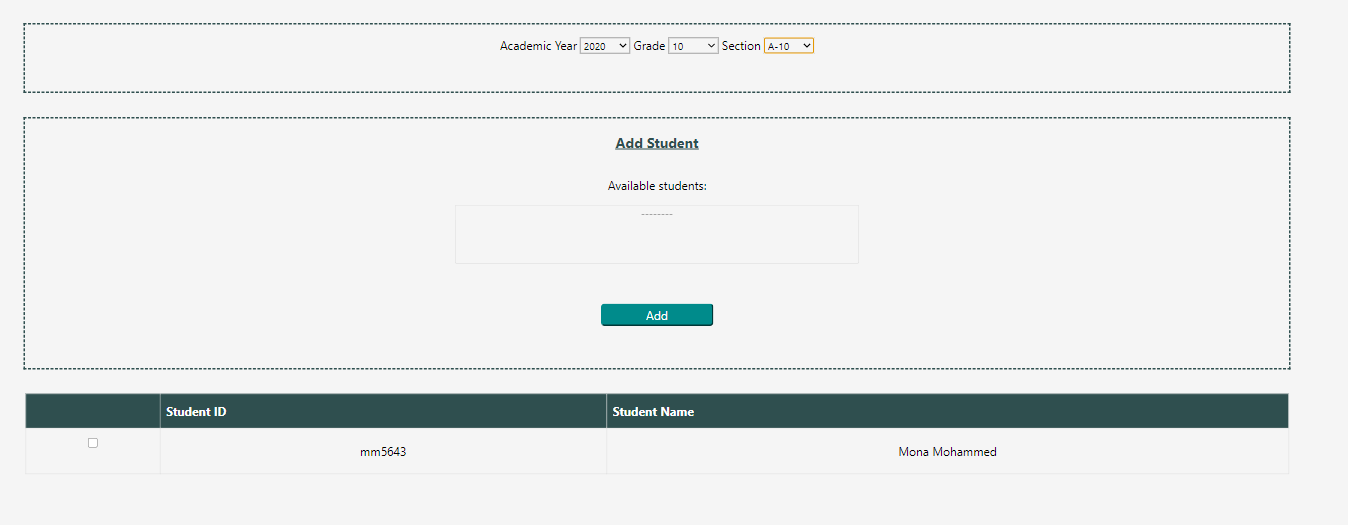
Delete:





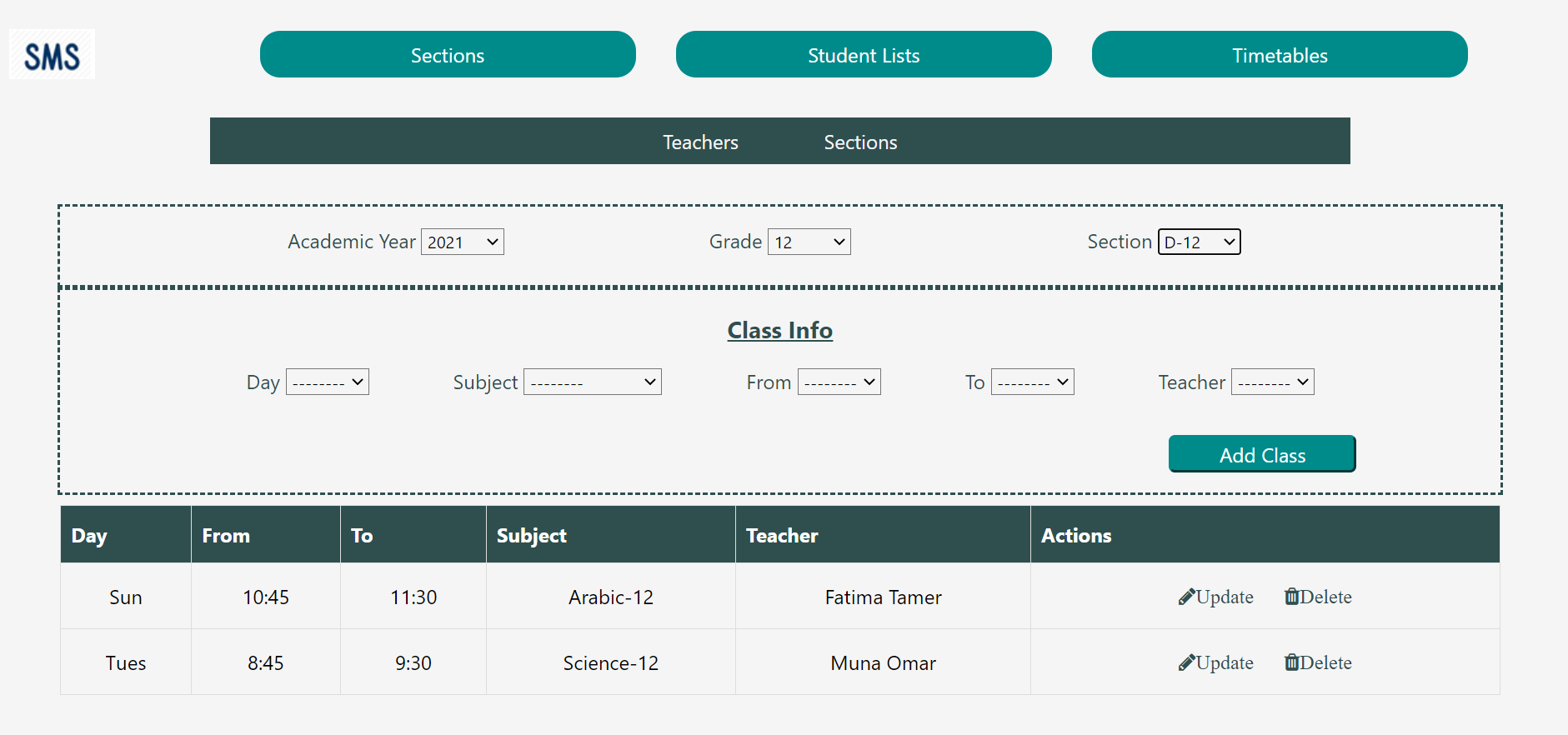


If the principal chooses to view students from old section, it will disable the adding and deleting:

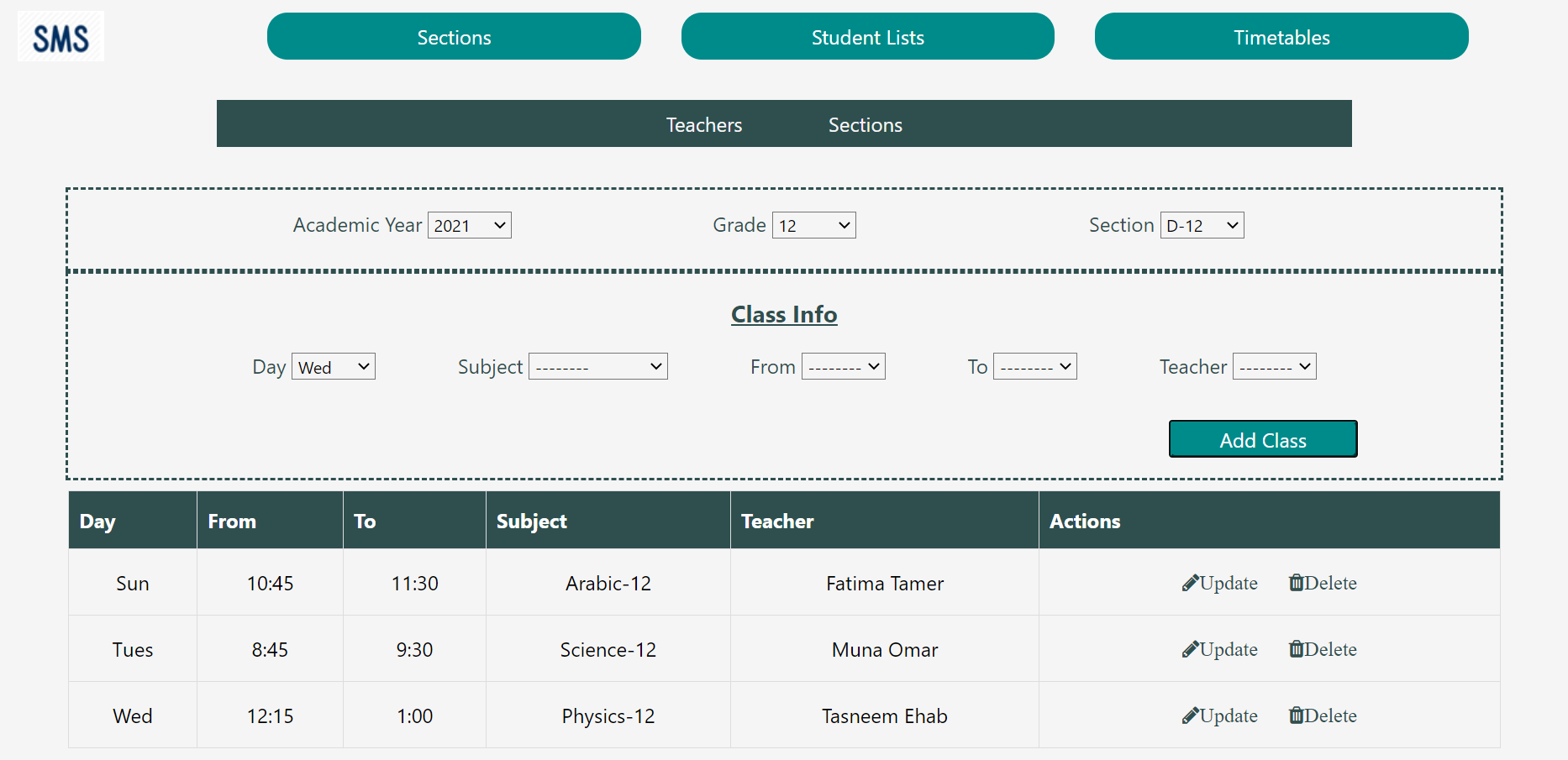


# Section courses (List, Add, Delete)

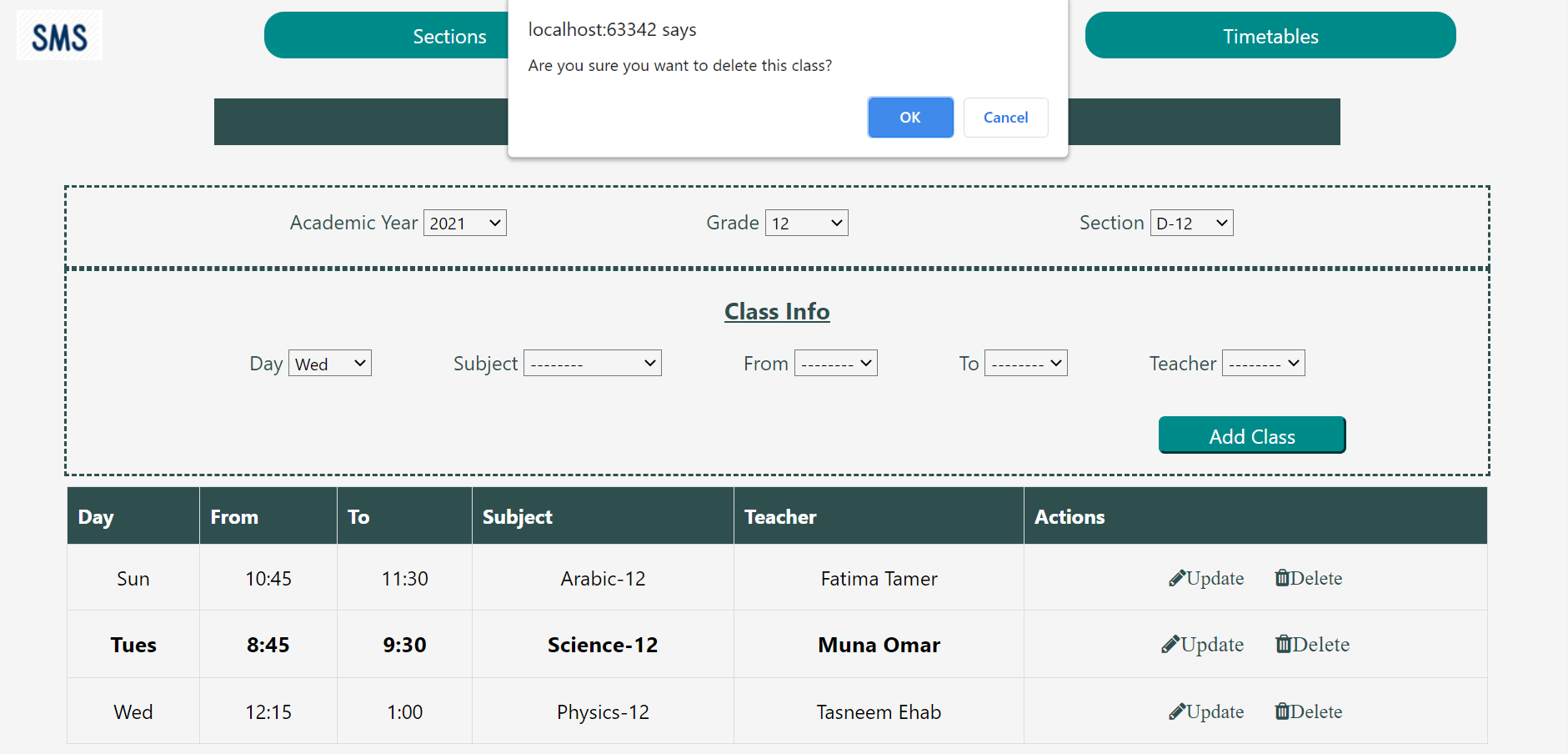
List:

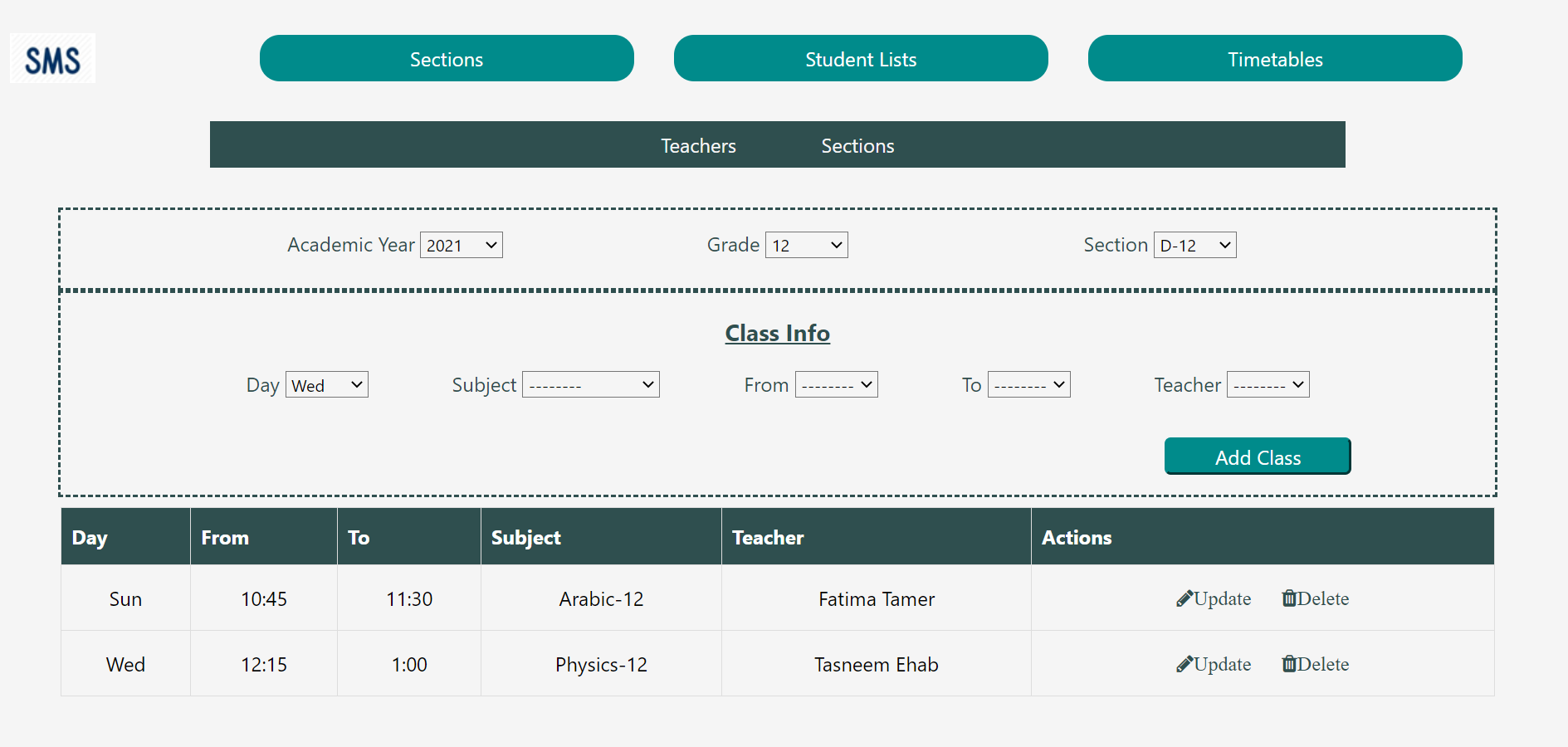


Add:

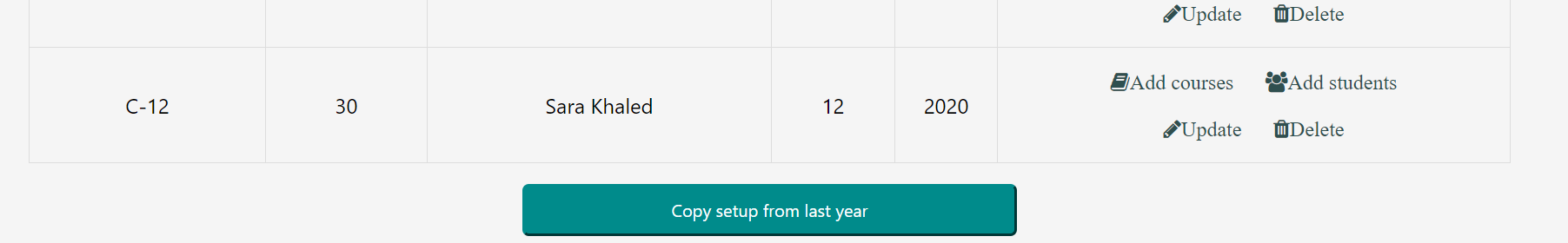


Delete:





# Copy sections setup from last year

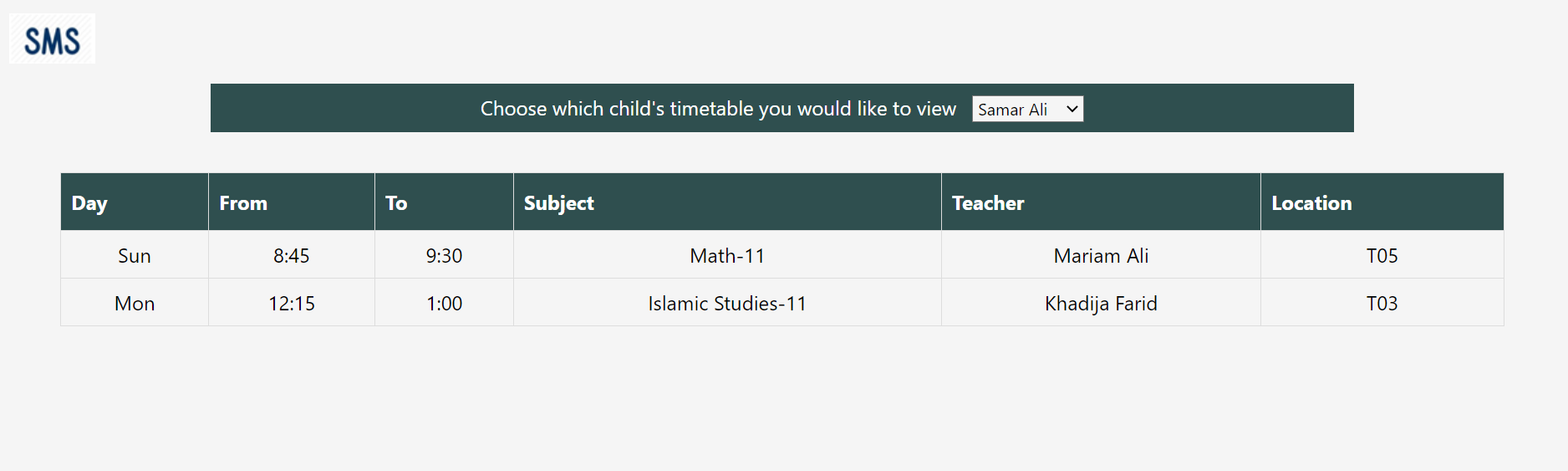


Section copied and the button becomes invisible because there are already sections in the current year:

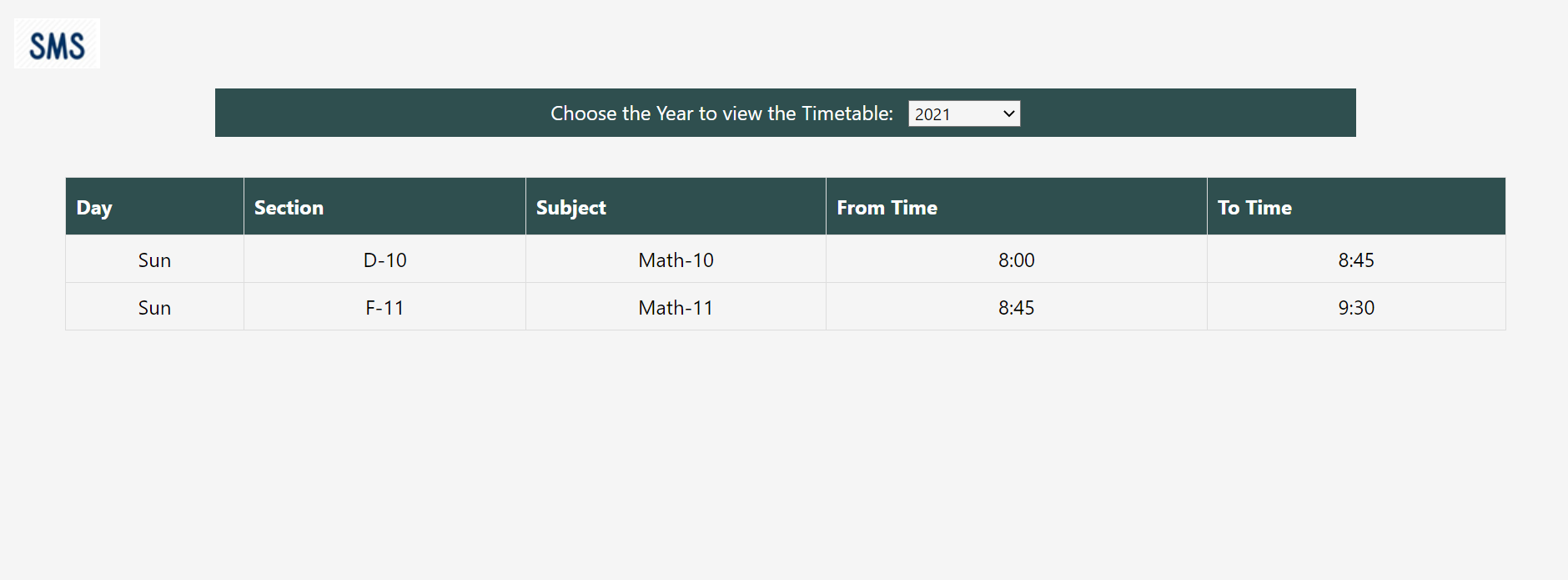


# Get timetable (Principal, Teacher, Parent)

Parent:

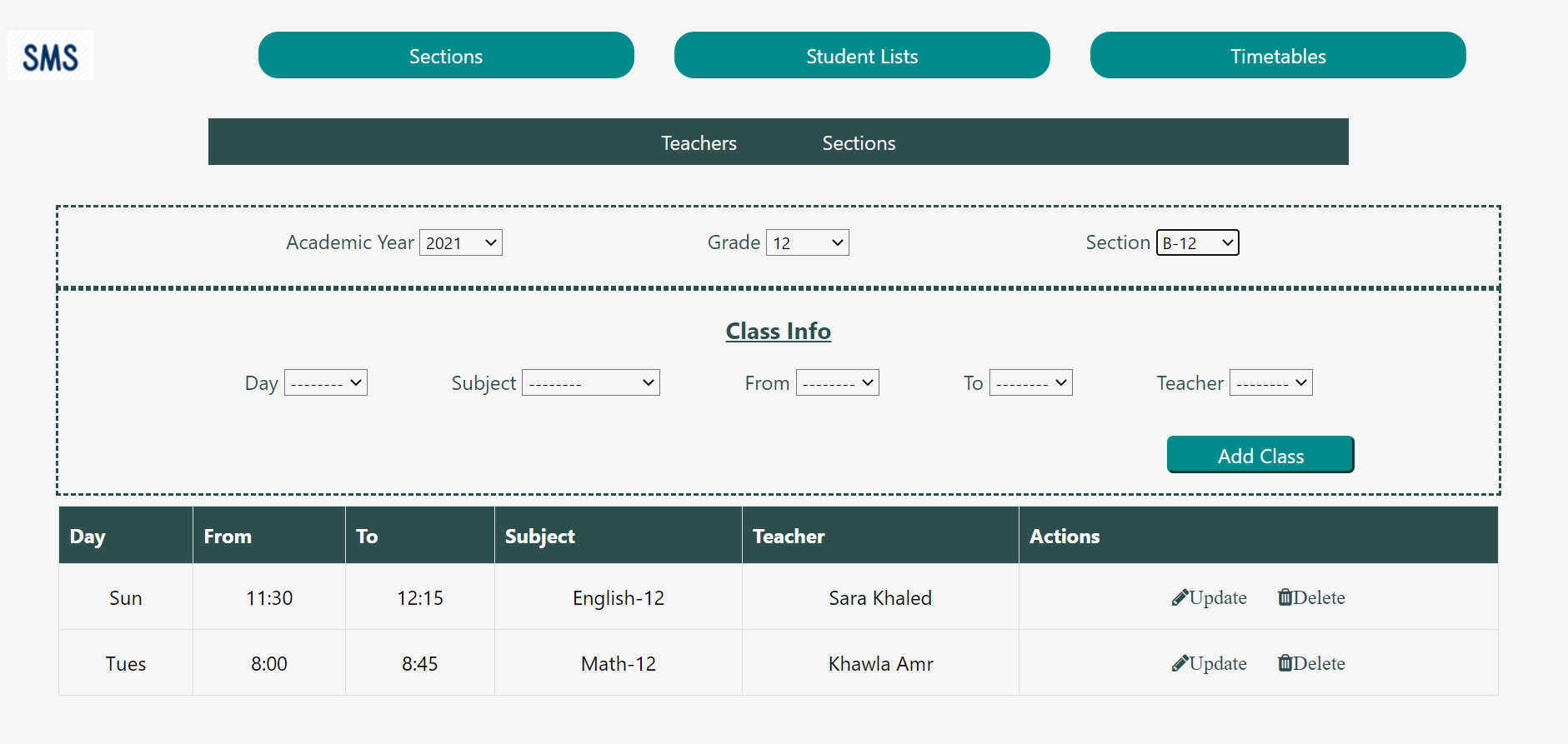


Teacher:

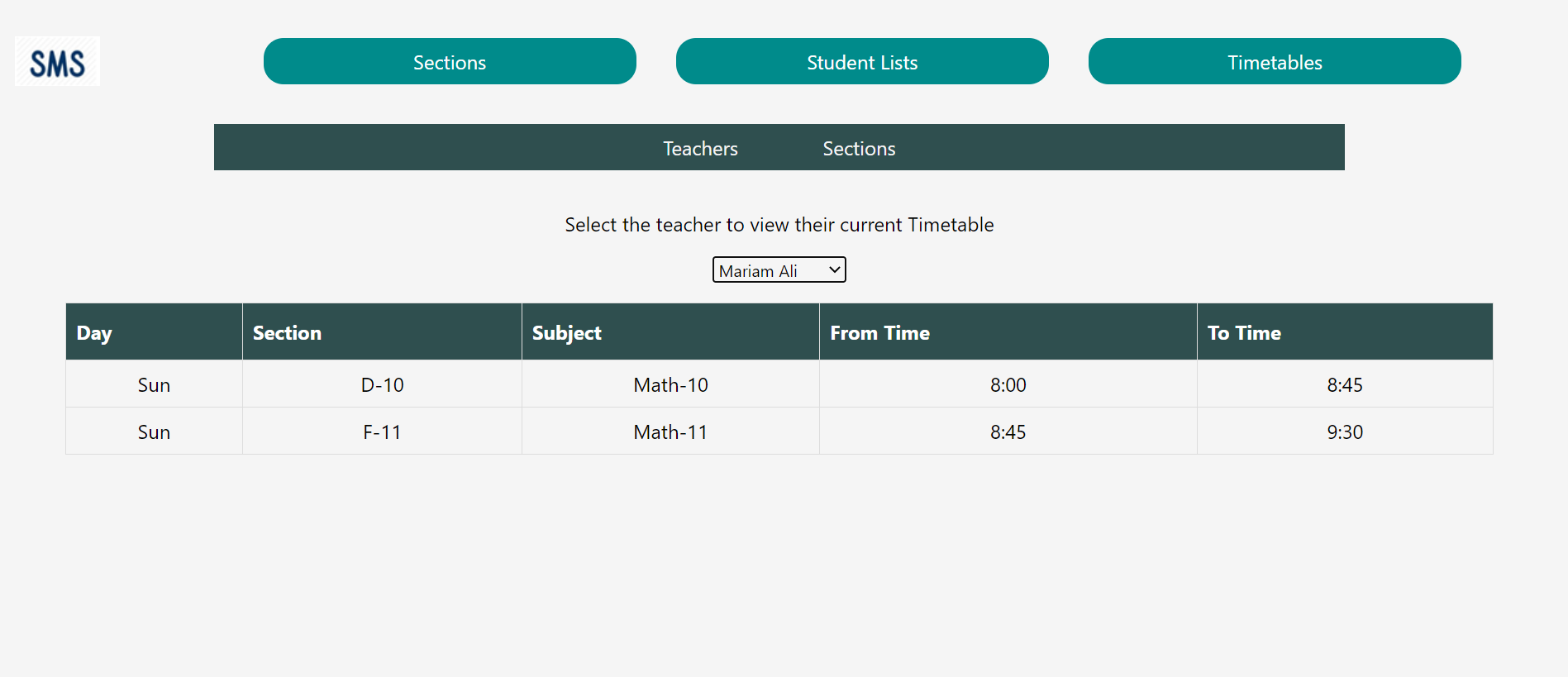


Principal for the sections (same as list courses):

For sections:



For teachers:



# Summary of team member contributions

Somaya:

I worked on implementing the Sections view for the Principal, allowing the principal to get the sections for a specific school year. I also implemented the ability to view the information about a specific section, including that the principal can see the basic information about the section, the list of section students and section courses. I also implemented the editing of the section info, and navigation to the section student list and section course list. Additionally, I allowed the principal to delete a specific section. All editing and deleting is validated to ensure it is only done on a current year section. Furthermore, I implemented functionality to copy the setup for the sections from the previous year. Finally, I also created any related methods in the IndexedDB repository and any related HTML and CSS, in addition to sample section data and contributed to assigning the students to different sections.

Muraam:

In this project, I mainly worked on the Timetable viewer for the different sections, and enabled the functionality of viewing the current schedule for a teacher, viewing the schedule of different sections in different years and for different grades. Additionally, I worked on the functionality of adding courses to a section, updating and deleting them as well, while validating that the principal does not edit the courses for a section from a previous year. I also allowed syncing of these alterations with the Teacher’s schedule. Furthermore, I worked on the interface for the teacher to view their timetable and for the parent to view their child or children’s timetables. I also contributed to the filtering of students list and implemented IndexedDB methods related to this part of the project, creating of the sample data, and any related HTML and CSS.

Duaa:

I worked on the student part of the project which is related to assigning students to section, get section students, and delete students from a section. Including their html, styling, and JavaScript pages. In addition to related database work with generating student and parent json object data.