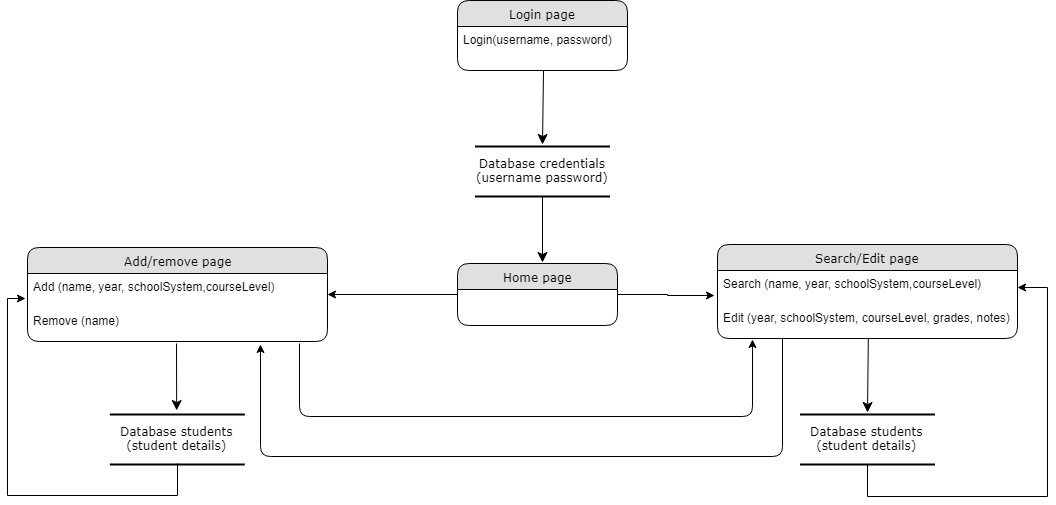
**Criterion B: Design**

**i) Design overview**

The application has a login page, a home page, a search/edit page and an add/remove page. Picture 1 illustrates how they will interact, and outlines how they will function. To create all the diagrams below, I used *draw.io* by *JGraph*.

Picture 1: Application’s main functionalities and interactions

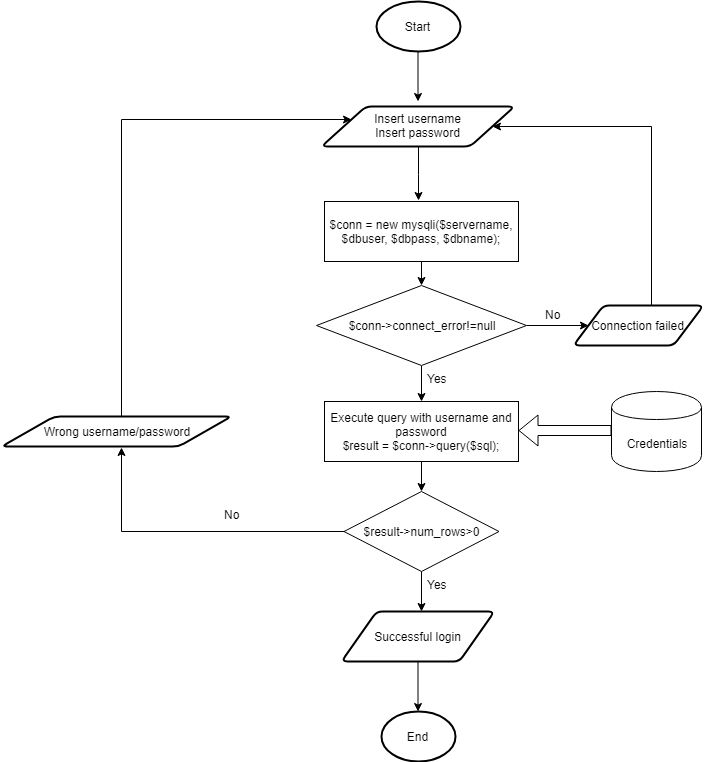


The application is designed to have 4 main pages, each one with a specific purpose (SC4).

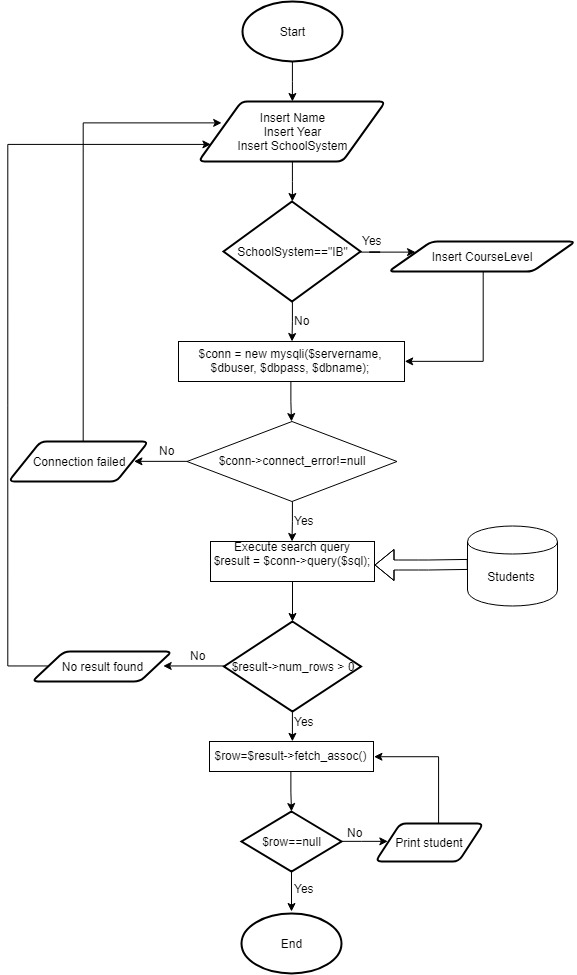
**ii) Functionalities description**

The pictures in the following pages are flowcharts that further demonstrate the four main functionalities of the application.

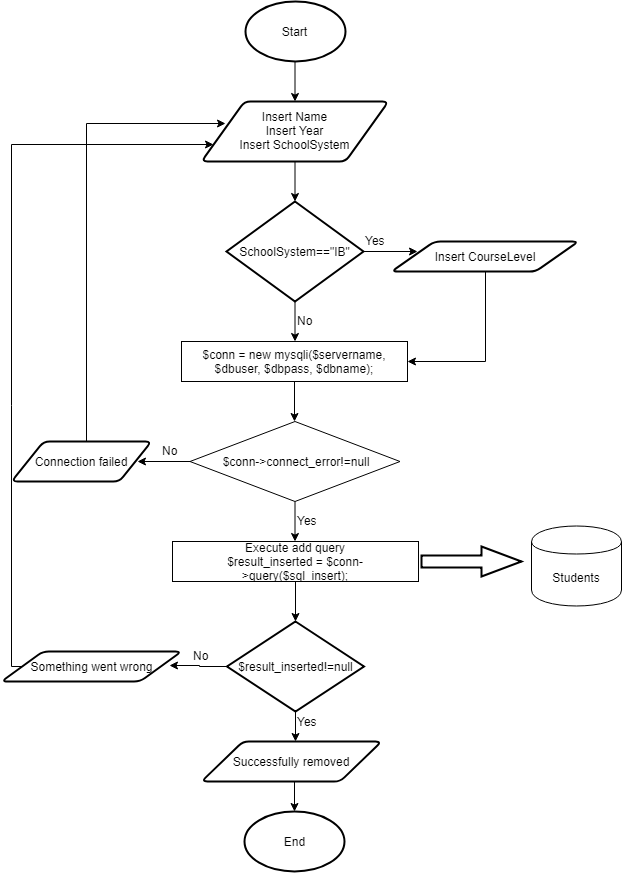
Picture 4: Flowchart for login functionality



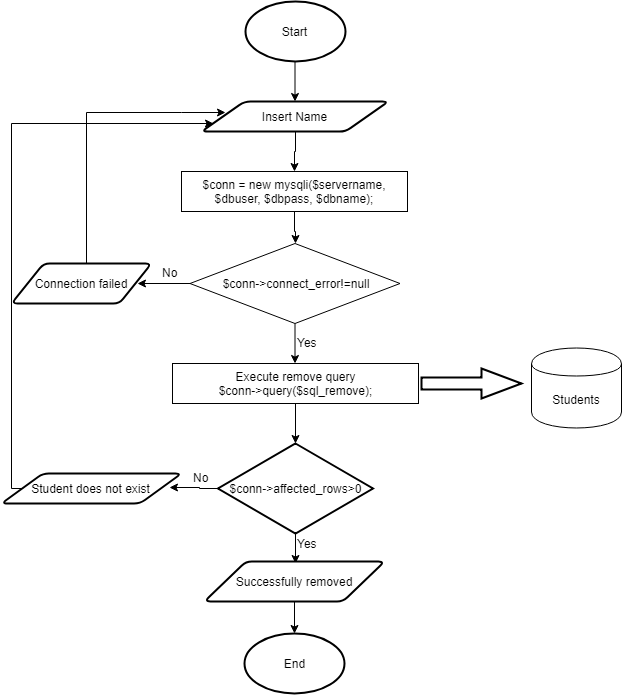
Picture 5: Flowchart for search students functionality



Picture 6: Flowchart for add student functionality



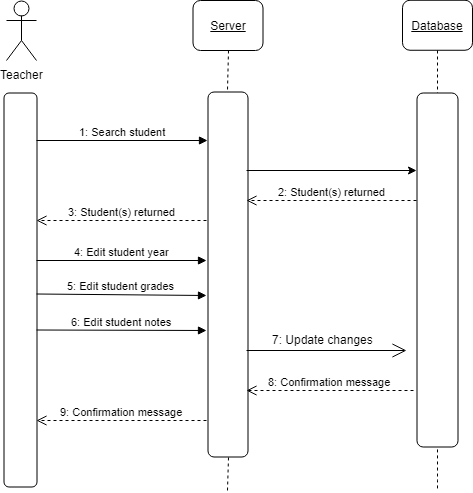
Picture 7: Flowchart for remove student functionality



**iii) Interaction with database**

In order to create this UML event diagram, I consulted the source *“What is Unified Modeling Language”* by *Lucidchart*. The following is a UML event diagram that illustrates the interactions between the user (teacher), the application and the database. Action 7 (Update changes) should take place asynchronously:

Picture 8: Interactions between teacher/server/database



**iv) Design of pages**

The following show the *design* of the pages, illustrating the desktop PC and mobile device versions (SC1).

Pictures 9-16 show the application’s *design* when accessed from a desktop PC.

Picture 9: The login screen

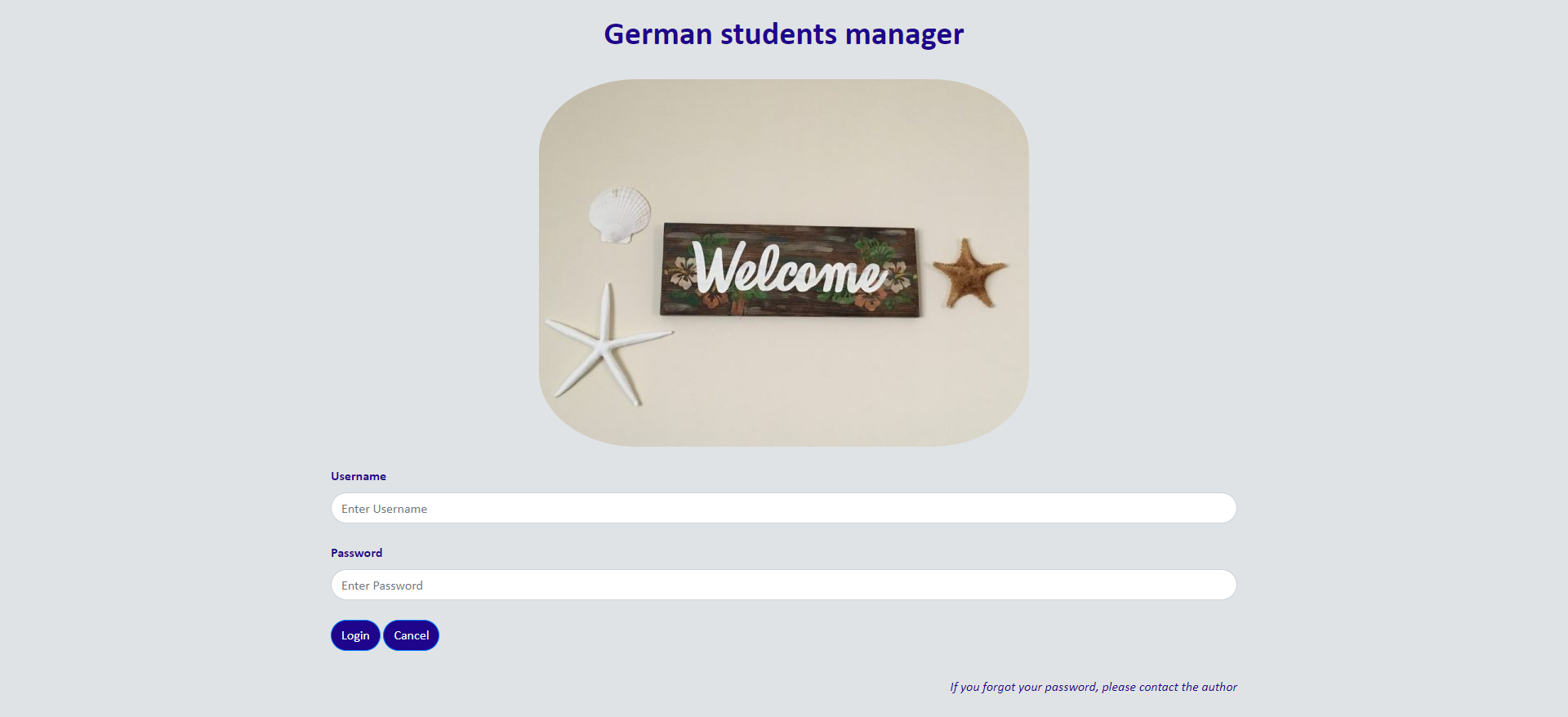


Image: “Tropical Rustic Welcome Sign with Hibiscus Flowers” (sea2landdesigns, n.d.)

Picture 10: The menu

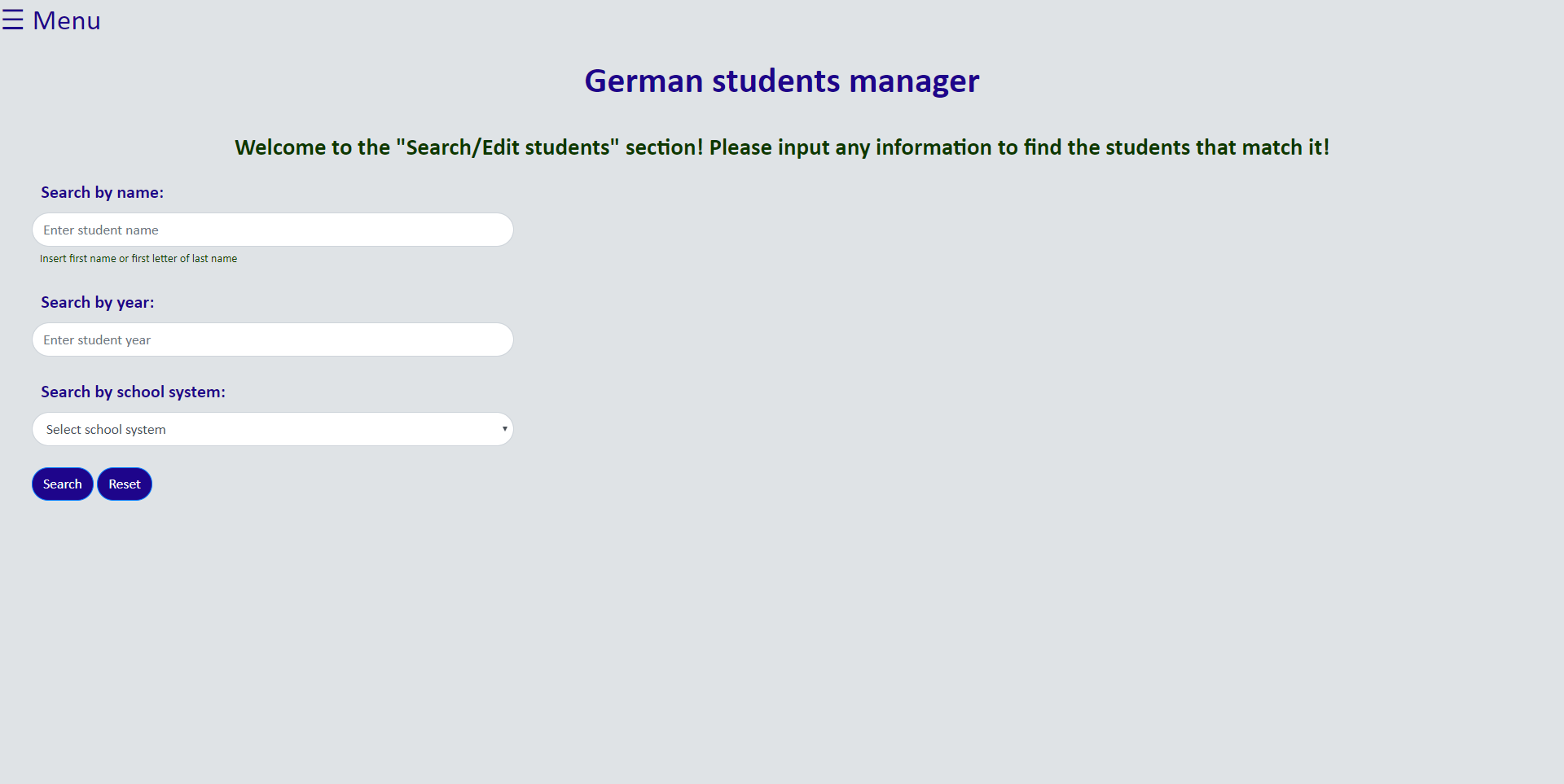


Picture 11: The home screen

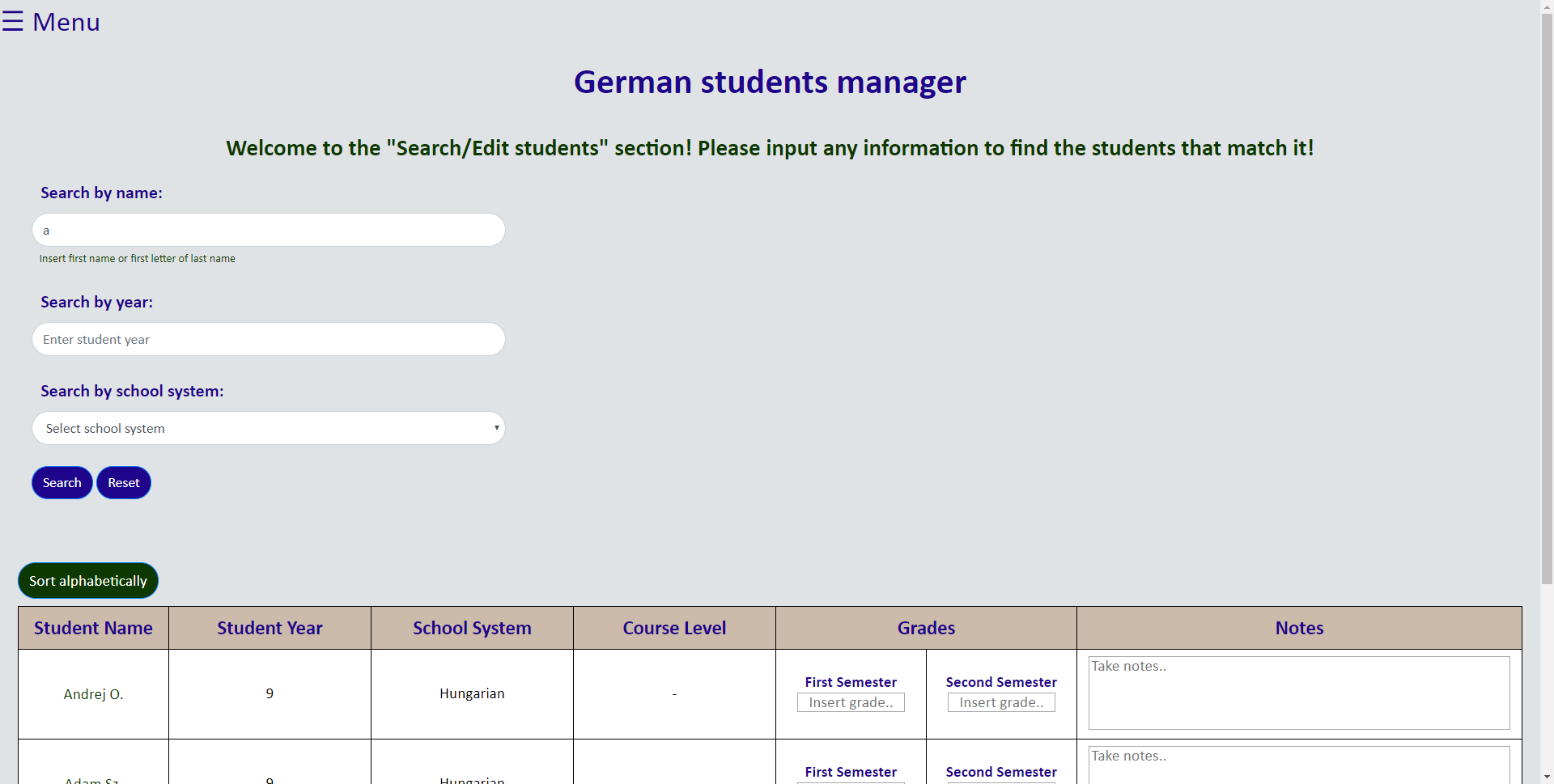


Image: “US Syria pullout could hurt fight against Islamic State: Germany” (Thinkstock, 2018)

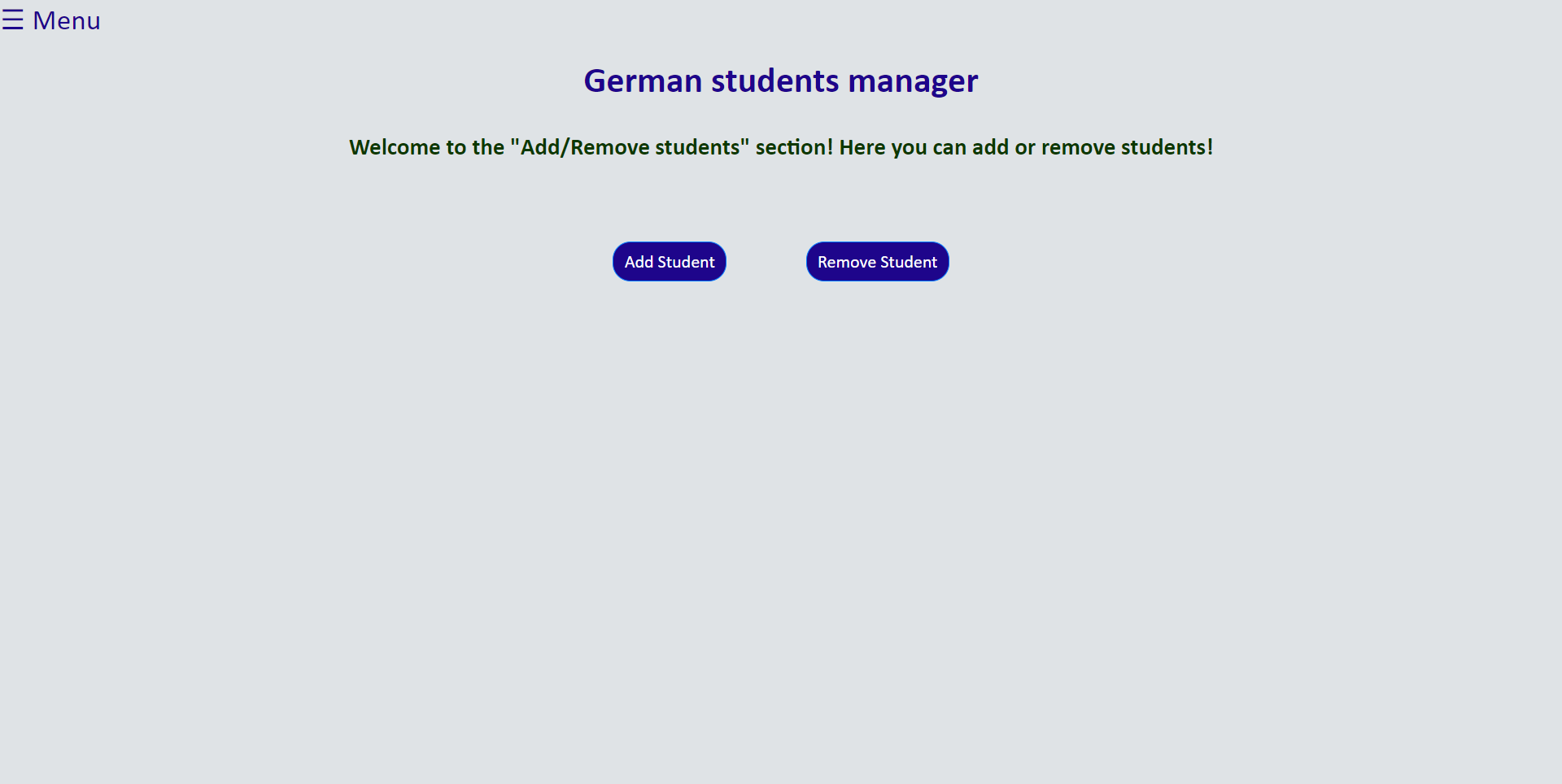
Picture 12: The search/edit students screen



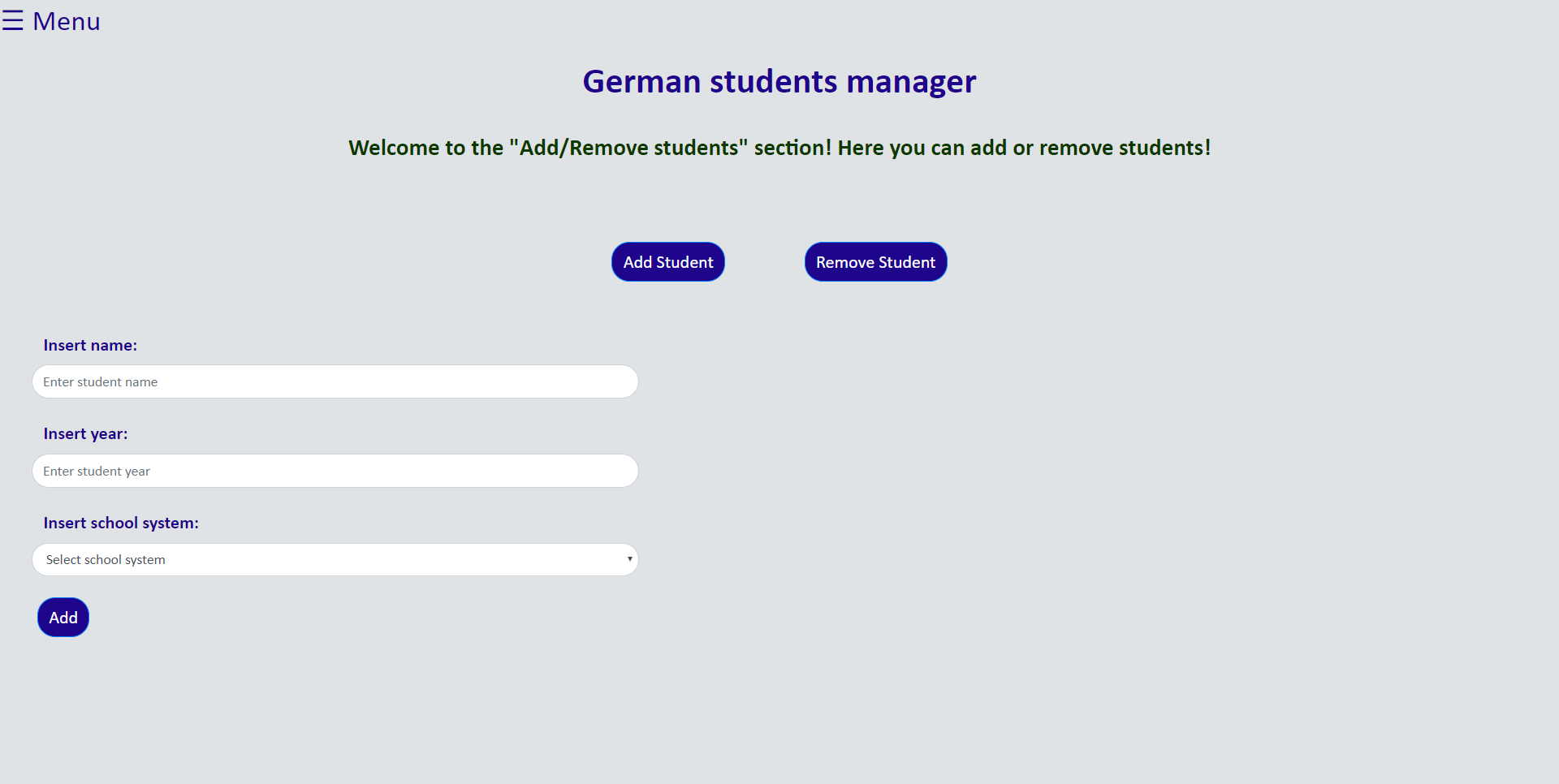
Picture 13: Expected design for search/edit students screen after search



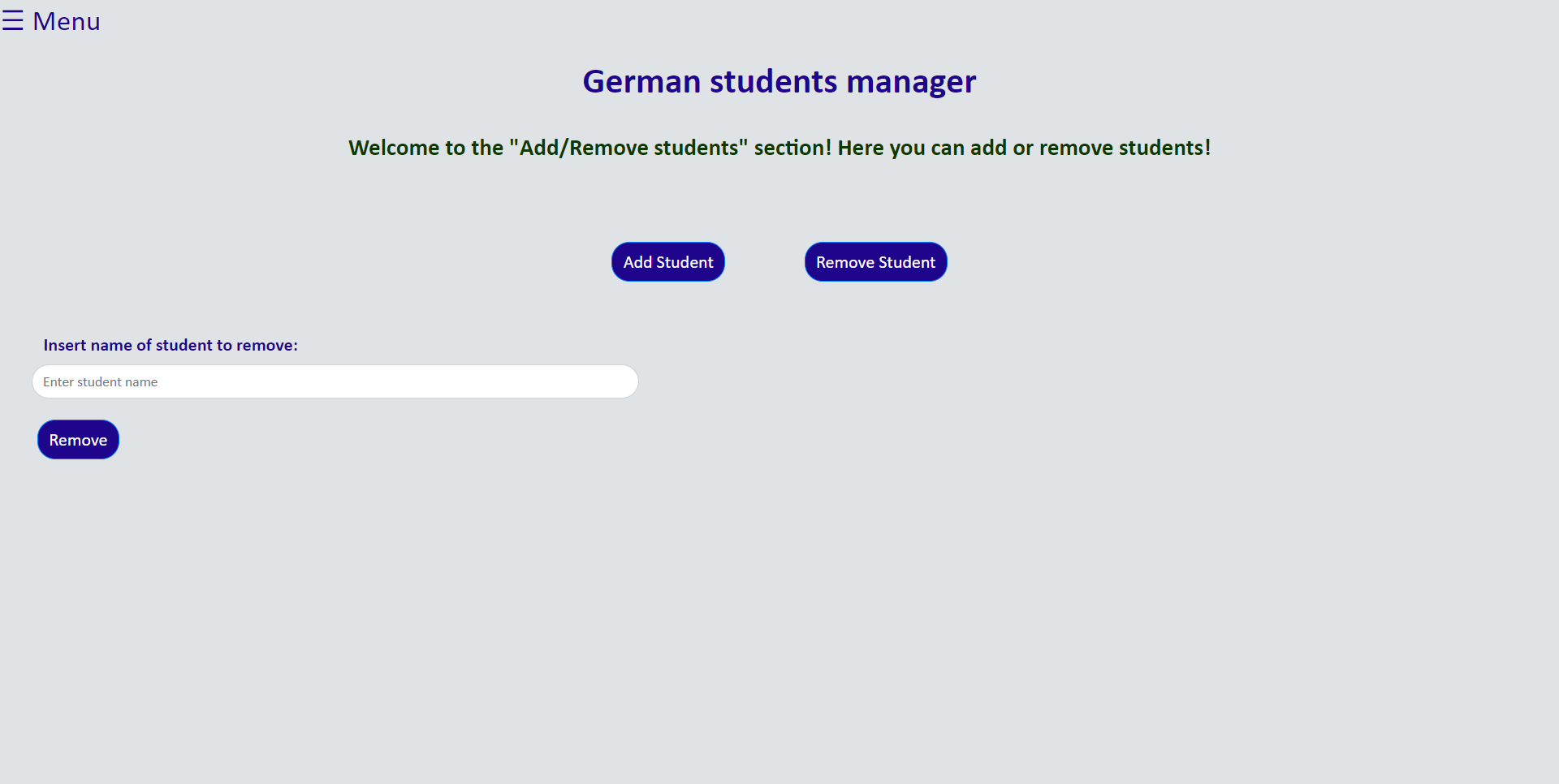
Picture 14: The add/remove students screen



Picture 15: The add student functionality

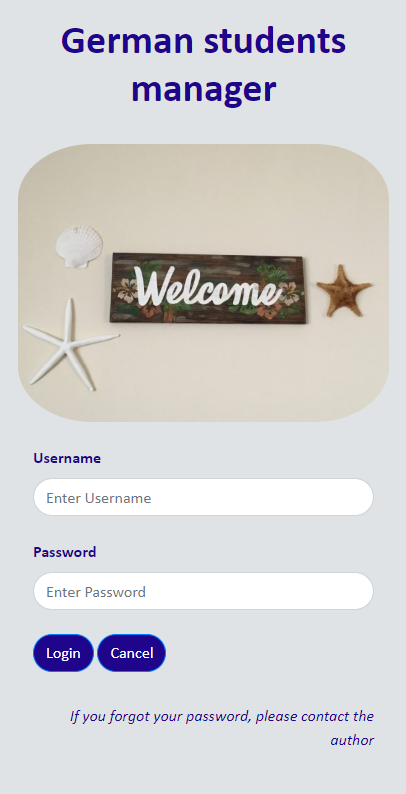


Picture 16: The remove student functionality

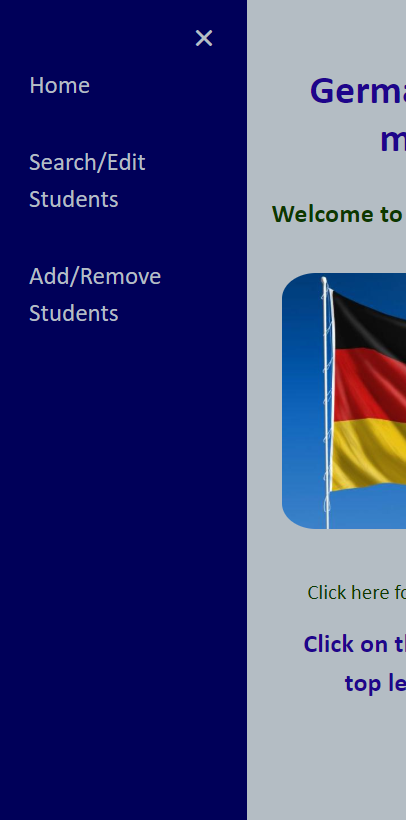


Pictures 17-24 show the application’s *design* when accessed from a mobile device.

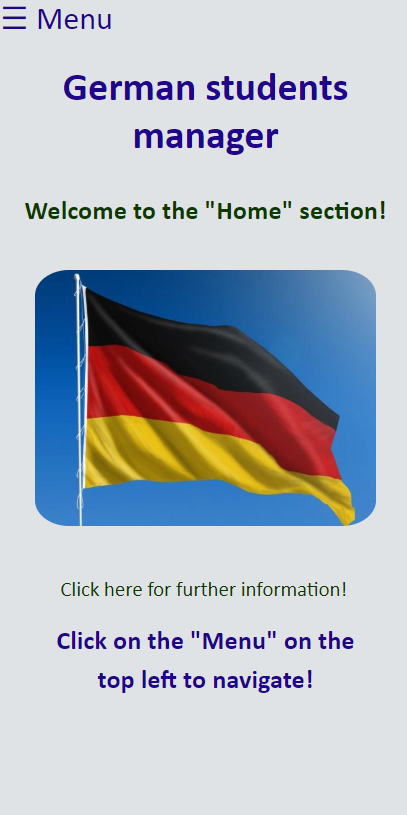
Picture 17: The login screen



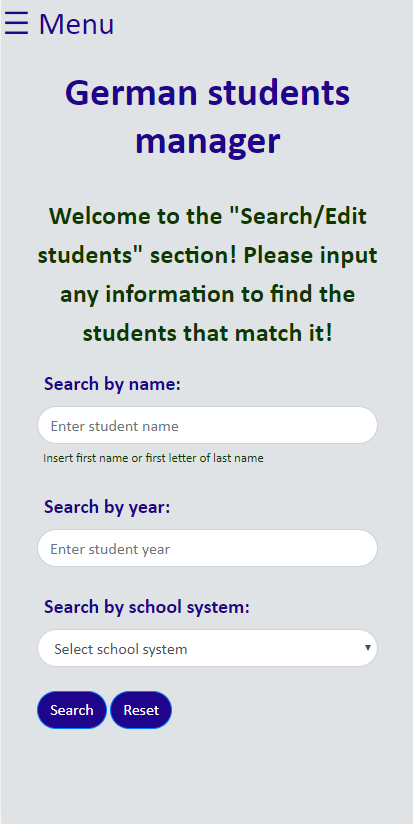
Picture 18: The menu



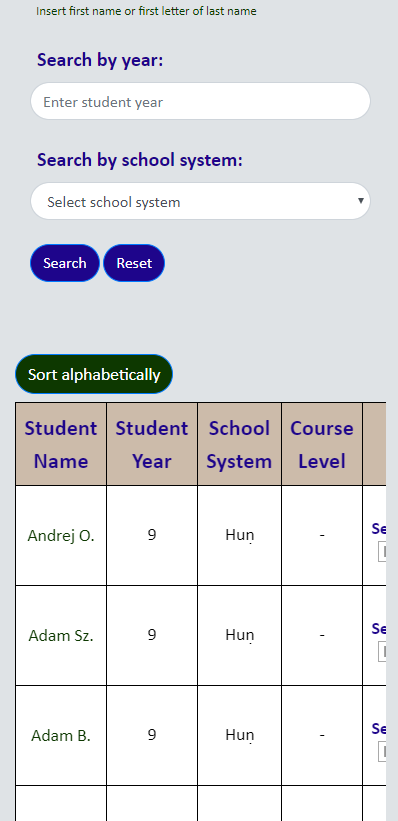
Picture 19: The home screen



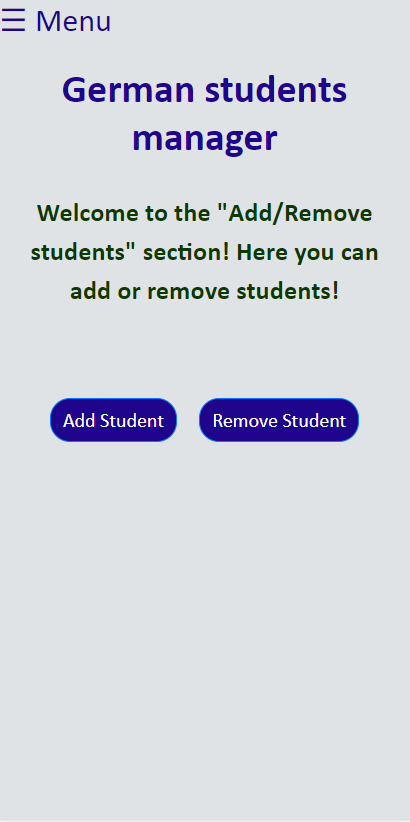
Picture 20: The search students screen



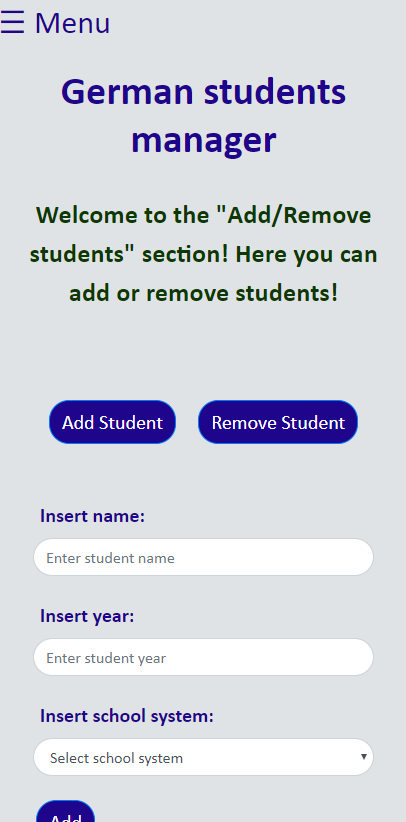
Picture 21: Expected design for students screen after search (the table is scrollable)



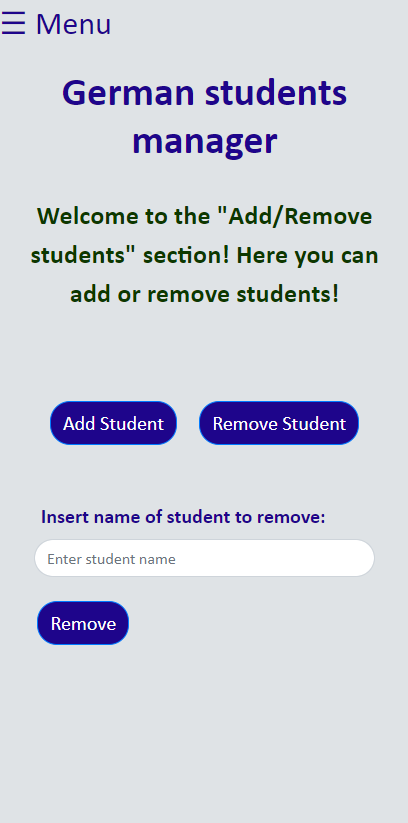
Picture 22: The add/remove students screen



Picture 23: The add student functionality



Picture 24: The remove student functionality



**v) Test plan**

|  |  |
| --- | --- |
| **Action to be tested** | **Test method** |
| The application can run | Open a web browser and visit the URL of the application |
| Login is forbidden if wrong username and/or password are inputted and notify the user (SC2, 8) | Input unmatching/incorrect username and password |
| Login is successful if correct username and password are inputted | Input correct username and password |
| The menu sidebar works properly (SC4) | Click on the “menu” button on the top left |
| The hints message opens when clicked on, and disappears when clicked again (SC5) | Click on the “Click here for further information!” sentence, then click on it again |
| The “Course Level” option in the search students section appears only while the school system is “IB” | Select “IB” as school system |
| The search button is disabled while invalid input is inputted in the name and/or the year search inputs and the user is informed why (SC8) | Put invalid inputs for either/both the student’s name or year |
| Searching is not allowed if all inputs are left empty | Leave all inputs empty and press search |
| The table appears when the search button is pressed, showing correct results according to given combinations (SC6) | Make a number of search combinations and make sure that the results are what expected |
| Everything on the table is editable (apart for the student’s name) | Try to edit the student year and put a letter, or try to edit the school system or the course level and put a number |
| Everything resets when the reset button is clicked | Click on the reset button after having searched for students |
| When a valid change is made on a student record, the database is updated and the user knows this happened (SC7, 9) | Edit a student record |
| The functionality to add or remove a student in the add/remove screen appears only after the respective button is clicked | Click on the add and then on the remove button |
| A student cannot be added if at least one input is empty | Leave one or more inputs empty and try to add/remove a student |
| When a student is added/removed through the application, the database is updated (SC10, 11) | Add/remove a student and check the database to make sure that the action happened correctly |
| The add button is disabled while invalid input is inputted in the name and/or the year inputs and the user is informed (SC8) | Put invalid inputs for either/both the student’s name or year |
| The remove button is disabled while invalid input is inputted in the name input and the user is informed (SC8) | Put invalid inputs for the student’s name |
| If something goes wrong while adding/removing a student (e.g. the student does not exist), the user is notified on what went wrong (SC8) | Try to remove a student that does not exist |
| When a student is added or removed, the user is informed (SC7) | Add/remove a student |