SQLite is a lightweight database tool that stores the result in a single file. However powerful, the tool does not comes with a visual schema editor, which limits the usability to more experienced users. The aim of the project is to implement a module that will assist in the process of database creation that will minimize the DDL SQL syntax knowledge requirement from the end-user. The tool should visualize both tables and relations between them. The tables, fields and relations names should be fully customizable. End user should be able to easily switch between modes that would allow to modify the tables, fields and relations.

There are two main functions of the each tool/editor module:
1) displaying the edited document (or more precisely – its part, as entire buffer can be to large to be displayed in the terminal) to the end user
2) handling events sent by end user to control the text document

In case of sql schema editor module end-user will operate on a sqlite database so the “document” from the point 1) is the sqlite database file. To ease the implementation of the specific tools, we will abstract handling of the physical signals from end-user by the backend modules. Backend module will provide an interface introducing unification in communication between end-user and the tool. The tool, in this particular case sqlite schema editor, needs to bind handlers of a given events to abstractly defined (using a string in a given format) signals. Handlers are nothing more nothing less than a functions that implement given functionality. SQLite schema editor should implement following handlers:

- to open existing sqlite database file for schema edition (treat the file name as binding parameter**)
- to write the buffer of currently edited schema into the user defined path (treat the file name as binding parameter**, if the path is empty schema should be written into the previously specified path)
- to switch between edition mode (tables/fields/relations)
- to jump between the items (either tables, or fields, or relations, dependently on the current mode)
- to select item to edition
- to handle all keyboard characters to perform edition of selected item (treat the value of the key as a KEY entry**)

The handlers should be bound for all selected* backend modules and should be sufficiently documented in the help guidance string of the handler binding. To bind all described handlers to all selected* backends properly read carefully the specification of their modules. You are free to choose binding sequence** for handler as long as you ensure the functionality will be available in all selected* backend modules (have, however, in mind usability aspect for selected binding sequence**).

The display window of the SQLite schema editor should **not** cover the first and the last line of the terminal – the lines should be reserved for backend modules purposes. The size of the display window should be dynamically adjusted to the terminal size. The display window should allow to scroll over the schema by arrow key pressed handle routine in a way dependent on the selected edition mode.

All aspects uncovered by the specification (or specifications of selected* backends) should be regulated by intergroup settlements or if that fails – joint group-teacher decisions.

* selected by one of the other members of your group, if none is selected – vim backend module should be considered as selected, **and should be implemented by entire group**
** see the specifications of selected* backends