Bash basics. Home task

1. **Create** an empty directory **/data** with file named ***users.db*** in the project directory that you used in previous home tasks.
2. **Create** a directory **/scripts** and a shell script named ***db.sh*** in this directory.

   This script must **support** the following commands:

   > db.sh add

   Adds *a new line* to the ***users.db***. Script must prompt user to type a username of new entity. After entering username, user must be prompted to type a role.

   Validation rules:

   username – *Latin letters only*

   role – *Latin letters only*

   New entity of ***users.db*** should be a comma separated value like: **username**, **role**

   Script must check existence of ***users.db*** file (for all commands accept **> db.sh** or **> db.sh help** ones) and prompt to confirm to create one if it does not exist and to continue initial operation after creation is completed.

   > db.sh

   Or

   > db.sh help

   Prints instructions *how to use this script* with description of *all available* commands (add, backup, find, list)

> db.sh backup

Creates a *new file*, named ***%date%-users.db.backup*** which is a *copy* of current ***users.db***

> db.sh restore

Takes last created backup file and replaces ***users.db*** with it.
If there are no backups - script should print: “**No backup file found**”

> db.sh find

Prompts user to type a *username*, then prints *username* and *role* if such exists in ***users.db***. If there is no user with selected username, script must print: “**User not found**”. If there is more than one user with such username, print all found entries.

> db.sh list

Prints contents of users.db in format: ***N. username, role***

where ***N*** – a line number of an actual record

Accepts an additional optional parameter **inverse** which allows to get result in an opposite order – from bottom to top. Running the command **> db.sh list inverse** will return the result as follows:

10. John, admin

9. Valerie, user

8. Ghost, guest

…

3. **Create** a shell script **build-client.sh** in **/scripts** folder. This script must invoke client app’s build command. When build is finished script must compress all built content/files in one **client-app.zip** file in the same **/dist** folder. Script must check if file **client-app.zip** exists before build and remove it from file

system. Script must use **ENV_CONFIGURATION** env variable to specify app's configuration to build.

NOTES:

It is highly recommended to use GitFlow WorkFlow to deliver changes into your project.

We suggest making conventional commits and commit linting in every app and repository, you work with during this course. You can use Husky npm package to setup appropriate git hooks in your project.

Please, share an access to your repository with your mentor and other students in your group.

When task is done, submit a pull request (PR) and request review from your mentor and other mentees from your group.

Duplicate PR's link and attach it in Learn Portal when you submit your work for review.

It is recommended to attach screenshots of your work/app along with PR's link (if any).

Notify your mentor and other mentees from your group when work is done and ready for review.

Evaluation criteria:

0 - Nothing has been done (obvious).
3 - Script **db.sh** is created and exists in the project's directory. Some (one or more, but not all) of its commands implemented and can be executed with issues.
4 - All commands of **db.sh** were implemented, but script or some commands work with issues. **build-client.sh** has not been implemented.
5 - All commands of **db.sh** and **build-client.sh** work as described above without issues.