AGH – UNIVERSITY OF SCIENCE AND TECHNOLOGY

Project documentation for

Bug Tracking System

Object-oriented programming languages

Electronics and Telecommunication EN, III year

Adrian Smoła

lecturer: Rafał Frączek

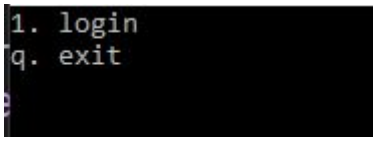
26.01.2021

1. Project description

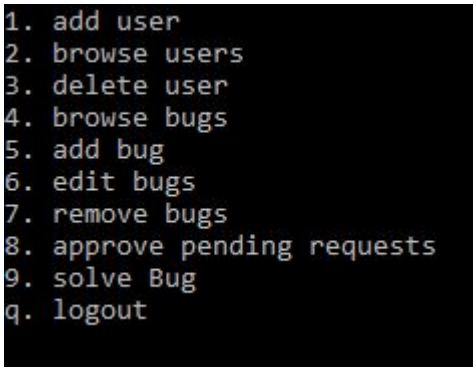
The topic of the project is the Bug Tracking System. I chose this project, as it sounded quite interesting to me. It splits the users into two categories: normal users and admins. Normal users can add, edit, and browse bugs, while admins additionally can manage all accounts, and approve bugs.

2. User's manual

When we run the program, we can see the base menu:

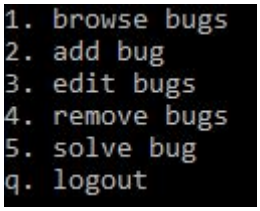


When we press the 1 button, we are taken to the login panel, where we enter login and password. After inserting correct credentials, we are taken to either the admin menu or user menu.



In Admin menu above, we have the following sub-menus:

- Add user - allows us to add new users;
- Browse users - Displays currently created users
- Delete user - Allows us to delete the account of specified user
- Browse bugs - displays bugs, with additional functions of displaying only a single category of bugs or only bugs that weren't approved
- Add bug - allows us to add a new bug to the database
- Edit bug - allows us to edit existing bugs
- Remove bugs - Allows us to remove the bugs from the database
- Approve pending requests - Admins can approve bugs created by users
- Solve bug - allows us to set the bug as solved
- logout - returns to the initial menu, allows us to login again, possibly to different account



In User menu above, we have the following sub-menus:

- Browse bugs - displays bugs, with additional functions of displaying only a single category of bugs or only bugs that weren't approved
- Add bug - allows us to add a new bug to the database
- Edit bug - allows us to edit existing bugs

- Remove bugs - Allows us to remove the bugs from the database
- Solve bug - allows us to set the bug as solved
- logout - returns to the initial menu, allows us to login again, possibly to different account

3. Compilation

In order to run the project, it's recommended to use the Visual Studio 2019, or simply run the provided exe file.

4. Source files

The project consists of the following source files:

- *main.cpp* – Main file invoking interface
- *header.h* – File with declaration of different classes
- *interface.cpp* – Here is the interface class, responsible for the interface of the entire project
- SQL_services.cpp - Here is a class containing all the methods used to communicate with the database
- sqlite3.h - Required in order for sqlite to work properly

5. Dependencies

The following external libraries are used in the project:

- SQLite – database SQL engine. Website: https://sqlite.org/.

6. Class description

In the project the following classes were created:

- `Interface` – Responsible for the running of the program
  - `void loginMenu()` – Displays the login menu
  - `void login()` – responsible for logging in
  - void adminMenu() - Displays admin menu
  - void usrMenu() - displays user menu
  - void addUser() - responsible for displaying a menu to add users
  - void browse(string _database) - displays additional options for browsing allows us to browse both bugs and users
  - void addBug() - menu to add bugs
  - void approveReq() - menu to approve bugs
  - void editBug() - shows menu to edit existing bugs

- void remove(string _database) - is used to delete either users or bugs from database
- void returnToMenu() - simple function to return to main menu, depending on the logged in user
- void wrongInput() - simple function to display that we have incorrect input
- void bugSolve() - allows us to set the bug report as solved

- `SQL_Services` – a class that is responsible for communication with database
  - `void add(string _perm, string _login, string _password, int a)` – Function responsible for adding users to the database
  - `int checkLogin(string _login, string _password)` – Checks whether the credentials are correct
  - void add(string _dev, string _desc, string _bugtype) - overloaded functions for adding bugs to the database
  - void browse(string _database, char _type, string _parameter) - overloaded function, taking either one, two or three arguments depending on the query, used to display records from the database
  - void approveBug(char _a) - used to approve bugs in the database
  - void remove(char _a, string _database) - used to remove users or bugs
  - void editBug(char _a, string _update, string _sqltype) - used to edit bugs
  - void bugSolve(char _a, char _b) - used to set the bug as either solved, or not solved

7. Resources

In the project the following resources are used:

- `myDBcp.db` – SQL database with following tables:
  - USERS
    - USER_ID - unique ID
    - ROLE - determines whether user is an admin or an user
    - LOGIN - stores accounts login
    - PASSWORD - stores account password
  - BUGS
    - BUG_ID - unique ID
    - APPROVAL - whether the bug is approved or not
    - DEV - who reported the bug
    - TYPE - What is the type of the bug
    - DESC - Description of the bug
    - SOLVED - whether the bug is solved or not

8. Future development

As for additional functionalities, we could implement additional roles in the team, for example manager, to manage the bugs, but not able to manage the users. We could also implement a more robust verification for logging it.

9. Other

predefined accounts:

- login: *admin* password: *admin*
- login: *user* password: *user*