**Ticket System Program User Manual**  
**(Glitch Hunter)**

**Project Description:**

The Glitch Hunter program is a ticket tracking system that is designed to work for internal use of technical teams. The tool can help technical teams keep track of items they are working on and projects they are working on. In addition, it can help managers keep track of the state of existing projects or tickets.

**Project Disclaimer:**

This program is a prototype to describe the functionality of a ticketing system. This is the first step into generating a final product. Let’s keep in mind that this is proof of concept and does not consider security, stability or other concepts that would be fundamental for a final product. It is expected that python 3.7 will be installed on the local machine. Currently the program is built for windows 10, we cannot ensure that the software will work at this point for Unix systems.

**Use case configuration:**

The program was created using python and python packages and dependencies. It is expected that python 3.7 will be installed on the local machine. Currently the program is built for windows 10. Note that in case of using some IDEs such as visual studio code, the correct interpreter must be selected, and the correct extensions to work with python must be installed, otherwise the program will not work as intended.

The following dependencies need to be installed in the local environment.

flask>=2.2.1,<2.2.3

flask\_wtf>=0.14.2,<1.0.0

WTForms>=2.2,<2.3.1

email-validator>=1.1.1,<1.2.2

sqlalchemy>=1.1.13,<1.4.39

Flask-Session

An easy way to install the dependencies is by running the setup.bat file. After installing the dependencies in the local environment, the local environment should be activated. This normally can be done while using the setup.bat file on windows.

Text

Description automatically generated

**Figure 1:** location of the setup.bat file on the project



**Figure 2:** run the setup.bat to setup the local environment.

notice that some antivirus will verify the workings of your program before allowing the execution, make sure that the antivirus does allow your program to execute. After the local environment is created, verify that the local environment is activated as shown in figure 3.   
  
Text

Description automatically generated

**Figure 3:** local environment is created and activated in the local IDE

In case that the program is not activating properly verify that you are using the command line and not power shell, and also verify that, your IDE is setup to use the correct environment, this process is show in figure 5. If all of this fails then go into the localEnv folder, then navigate to the Scripts folder and execute the activate.bat script, as shown in figure 4.

Text

Description automatically generated

**Figure 4:** activating the environment in case that the activation process fails.   
  
Graphical user interface, application, website

Description automatically generated  
**Figure 5:** IDE using the local environment interpreter for python 3.7

After the environment has been created and activated, then we will create the database. This can be done by navigating to the GlitchHunter folder and running the create database program or createdb.py program, this will generate the database. In that case a local sqlite database file will be created. The location of the file is shown in figure 6.

A screenshot of a computer screen

Description automatically generated with medium confidence

**Figure 6:** location of the createdb.py file

To run the application, the flaskapp.py program will be run using the activated local environment. Then the flask webserver will be activated and we will be able to see the webpage using a browser and typing localhost:5000

Note that to run the application your environment will have to be activated as shown in figure 7.

Text

Description automatically generated

**Figure 7:** running the flaskapp.py program

Graphical user interface, application

Description automatically generated

**Figure 8:** Running the page in localhost:5000

Use case login:

There is already a sample account included in the database of the application, the test user-name is demo and the test password is demo. If we input the correct user name and password we should be able to log into the application.

Graphical user interface, application

Description automatically generated  
**Figure9:** Login using the demo account

Use case tickets

The application gives us access to tickets in the database, we can also add new tickets into the database using the website.

Graphical user interface, application

Description automatically generated  
**Figure 10:** Ticket section of the project

Use case projects  
The application gives us access to the project in the database, we can also add new projects into the database using the website.

Graphical user interface, text, application

Description automatically generated

**Figure 11:** Project section of the program

Use Case Customer

The program in addition to keep track of tickets and projects it can also keep track of customer information, this can be useful to keep track of customer details.  
  
Graphical user interface, text, application

Description automatically generated

**Figure 12:** Customer section of the program