Contents

[**1.** **Introduction** 1](#_Toc133236431)

[**2.** **Overall Description** 1](#_Toc133236432)

[**3.** **System Design - UML** 1](#_Toc133236433)

[**4.** **Quality Considerations** 1](#_Toc133236434)

[**5.** **Testing Strategy and Proof** 1](#_Toc133236435)

[**6.** **Documentation** 1](#_Toc133236436)

[6.1. User 1](#_Toc133236437)

[6.2. Technical 1](#_Toc133236438)

[**7.** **Conclusions** 1](#_Toc133236439)

[**APPENDICES** 1](#_Toc133236440)

Tynan Orr and Spyridon Sakellariou

ITC Software Development

 Evgenia Vagianou

April 25, 2023

# **Introduction**

<<This code created is a ticketing system that allows players of a game to create tickets and for two employees to manage them>>

Keywords: json, user, employee, ticket

# **Overall Description**

<< This was a two-person project to create a ticketing system with a nice user interface and ticket, user and employee management functionalities, We decided to use python and not have a graphical user interface and to manage the information using json. This ticketing system is for reporting problems with SuperGalactic Racing Games!!! With categories for technical, financial or other issues, It also is username and password protected for each user >>

* 1. Requirements (as high-level list of features): The requirement for this project were for it to include two types of users with a username and password restricted login and a user-friendly console interface, with Ticket, Category, Player, and Employee management.
  2. User Classes and Characteristics

User class: tech literate, age of 12+

Employee class: tech literate, age of 18+

# **System Design – UML**

Diagram

Description automatically generated

# **Quality Considerations**

Performance: We focused on making sure the code works properly

Conformance: We tried to make the menu intuitive and general

Serviceability: We focused on making the code very self-explanatory so that it would be easy to edit

# **Testing Strategy and Proof**

Smoke Testing: Here are some of the tests.

# **Documentation**

## User: All the user has to do is follow the instructions displayed by user interface

## Technical: Utility classes in this code include json \_parser, filesystem\_creater, and generic\_functions

# **Conclusions**

The code works great things that could be added are an administrator user

# **APPENDICES**

Team reflections

It was much more difficult than expected to work together properly and there were a number of errors that required redoing much of the code.

Deployment instructions

Uses Python 3

The employees

usernames:user\_1 and user\_2

passwords:user1, user2