CST8333

Assignment 1

**Just Ticket**

Prepared By  
Joey Gilbert

2024-04-01

Algonquin College

All material prepared for this assignment was produced by the author. Material from all third parties has been cited and referenced.

Table of Contents

[Purpose 3](#_Toc162876781)

[Timeline 3](#_Toc162876782)

[Status 3](#_Toc162876783)

[Requirements 3](#_Toc162876784)

[Traceability Matrix 4](#_Toc162876785)

[Solutions Comparison and Selection 5](#_Toc162876786)

[Solution 1 – Standalone - Desktop Application 6](#_Toc162876787)

[Solution 2 – Web Application 6](#_Toc162876788)

[Selection 6](#_Toc162876789)

[Programming Languages Comparison and Selection 6](#_Toc162876790)

[Selection 8](#_Toc162876791)

[Conceptual Design 8](#_Toc162876792)

[Package Diagram 9](#_Toc162876793)

[Class Diagram 10](#_Toc162876794)

[Entity-Relationship Diagram 12](#_Toc162876795)

[UI Prototype 13](#_Toc162876796)

# Purpose

The Just Ticket system will be a web-based application designed for businesses to record work orders and provide support for their customer base. This system will have two unique roles of customer or agent where they will be generated a view based on their specific role. This ticketing system can be utilized for sales, billing, and or technical support enquires and can be design to the specific needs of said enterprise.

# Timeline

The timeline chart, below, shows project tasks and major project milestones. The blue extending from each point shows how much time has been allocated for task completion.

## Status

As at the time of reporting the project is tracking well with our prototype designed at roughly 60% completion. The code is still rough, but I will keep iterating through the requirements until I have an appropriate V1 to test.

**A blue and white timeline

Description automatically generated**

# Requirements

This project will meet a combination of Technical, Business, User and Non-Functional requirements. Through the Agile Iterative process requirements will be refined and scoped in and out through the SDLC lifecycle. A base requirement set will be decided and a baseline working prototype will be implemented. Further requirements will be grouped into a sprint backlog to be completed by priority and business need.

## Traceability Matrix

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Requ’t Number** | **Requ’t Name** | **Category** | **Test** | **Expected Result** | **Result** | **Pass/Fail** | **Notes** |
| **Criteria** |
| **TR-01** | MySQL Database | Technology | Connect to the Database | Database connects successfully | N/A | N/A | N/A |
| **TR-02** | Mysql Libraries | Technology | Import MySQL Libraries to IDE | Mysql Libraries Imported Successfully | N/A | N/A | N/A |
| **TR-03** | Python Flask | Technology | Install and run python flask | Python Flask works and displays hello world | N/A | N/A | N/A |
| **TR-04** | Login Page GUI | Technology | User Can view Login UI | Page loads and user can login | N/A | N/A | N/A |
| **TR-05** | User login | Technology | User enters username and password on login page | User Successfully logs into ticket system | N/A | N/A | N/A |
| **TR-07** | Ticketing System UI | Technology | User logins and records view | User is given view based on role information | N/A | N/A | N/A |
| **TR-08** | View Single Ticket UI | Technology | User clicks on hyperlinked ticket number | User is directed to a ticket detail page with ticket specifics | N/A | N/A | N/A |
| **TR-09** | Create Ticket UI | Technology | User attempts to create a new ticket | New Ticket is created and added to ticketing system database | N/A | N/A | N/A |
| **TR-10** | Users update ticket | Technology | User attempts to provide updates in ticket | Ticket updates are saved and all previous ticket updates are listed in ticketdetail | N/A | N/A | N/A |
| **BR-01** | Case Management | Business | Access Ticketing System and review tickets | Ticketing system provides a list of cases that are active | N/A | N/A | N/A |
| **BR-02** | Case Age | Business | Access Ticketing System and verify if cases have an age metric | Ticketing system shows list of cases and their age | N/A | N/A | N/A |
| **BR-03** | User level-based access | Business | Attempt to login via login page | View displayed depends on user role | N/A | N/A | N/A |
| **BR-04** | Reporting | Business | Access ticket system export feature | System allows an agent to pull an export of all cases for reporting purposes | N/A | N/A | N/A |
| **UR-01** | Instructions | User | Users are given an instruction document and proceed to use application | Instructions are clear and allow user to fully operate application | N/A | N/A | N/A |
| **UR-02** | Multiple Users | User | Users’ login and use application | Users are given a view based on their role | N/A | N/A | N/A |
| **UR-03** | Tickets Associated to Customer | User | Customer creates a ticket for agent review | Ticket is created and created by user is recorded | N/A | N/A | N/A |
| **UR-04** | Record Keeping | User | User actions need to be recorded and timestamped | User activity is clearly tracked in database and application | N/A | N/A | N/A |
| **NF-01** | Web app works in multiple browsers | Non-Functional | Access Web UI in Chrome, Edge, Brave | Works in Multiple browsers | N/A | N/A | N/A |

# Solutions Comparison and Selection

While conceptualizing this application there were two unique approaches. One was to make this as a standalone desktop application where the user could install this on a local server and utilize various networking methods to make this visible to users on an intranet.

Another approach was to develop this as a full web application that could be password protected via a login page. This would be accomplished by providing the necessary application files to install on a user’s domain, the application could then be used and managed on any webhosting platform.

## Solution 1 – Standalone - Desktop Application

This solution would be installed on a user’s desktop or server and be connected via an intranet so local user could then be use the application.

* 1. Could utilize Free Version of PyCharm
  2. Localized access
  3. MySQL Database for persistent data storage.
  4. Could use free libraries on Python such as Tkinter
  5. OS Specific design

## Solution 2 – Web Application

With this solution Users would simply navigate to a website that would have a login page. This would secure the application and allow users to be given a view based on user role. This would allow users and agents to connect from anywhere so a hybrid type work model that allows agents and customers to check on their tickets from any location.

1. Domain and hosting would be required, a VM could be purchases from Ovhcloud.ca for under 15 dollars a month, a domain could be purchased for under 20 dollars a month, <https://www.ovhcloud.com/en-ca/vps/>.
2. Free MySQL server databases could be created and hosted from the VM.
3. Licensing Fee would be required to utilized PyCharm with Python Flask
4. To manage networking and software requirements, application could be packaged in a Docker Container for ease of setup.

## Selection

Solution 2 is preferred over solution 1. Solution 2 instantly allows the application to be available to everyone on the web. This allows the application to be extremely portable over a standalone desktop solution. The added upfront costs would be easily absorbed by purchasing bare metal virtual machine servers that are under 15 dollars a month at ovhcloud.ca. A domain name is also very inexpensive and allows us to compete with the modern landscape of web applications.

Recent technologies like Docker Containers can be used to easily deploy this type of web application with all its necessary prerequisites. Utilizing this kind of technology can allow the application to be easily scaled depending on growth of the business using it.

# Programming Languages Comparison and Selection

Below is a simple chart I did comparing 4 popular languages. These languages are Python, C++, JavaScript, and Rust. The main factors I compared each against were syntax style, percentage of developers that say they use the language, common uses, Main Advantages, Main Disadvantages, Multiparadigm, average salary, and their available web frameworks. One thing to note is the salaries varied wildly from source to source, so in this analysis I chose a survey from (*Stack overflow developer survey 2021*) as this appears to have the more realistic salary values. Each statement for each language has been APA citated.

Programming Languages Evaluation

| **Criteria** | **Python** | **C++** | **JavaScript** | **Rust** |
| --- | --- | --- | --- | --- |
| Syntax Style | Uses New Lines, Whitespace, and indentation rather the semicolon and parenthesis. (Python introduction) | Uses curly brackets and semicolons for code structure (C++ introduction) | Borrows from C- style and utilizes semicolons and braces for code structure (What is JavaScript used for?) | Like other C++ style programs which utilize semicolons and curly brackets (Gavriluk, 2020) |
| % of developers that use language | 48.24% (Stack overflow developer survey 2021 Most Popular) | 24.31% (Stack overflow developer survey 2021 Most Popular) | 64.96% (Stack overflow developer survey 2021 Most Popular) | 7.03% (Stack overflow developer survey 2021 Most Popular) |
| Common Uses | Web Development, software development, mathematics, system scripting (Python introduction) | Operating Systems, Games, Embedded Systems, Software Library creation (C++ introduction) | Web development, Web applications, Web Servers, Mobile Applications (What is JavaScript used for?) | Embedded Systems, Robotics, Industrial Automation, Network Programming (Mittal, 2023) |
| Main Advantage | Easy to Read and Easy for beginners and multipurpose  (DuarTech Solutions, 2021) | Considered one of the fastest and most versatile languages (DuarTech Solutions, 2021) | Considered fastest and easiest code to implement on a browser (DuarTech Solutions, 2021) | Security with Memory Safety features (Tawfik, 2024) |
| Main Disadvantage | Interpreted Language resulting in slow execution time (DuarTech Solutions, 2021) | Not Ideal for web development (DuarTech Solutions, 2021) | Visible Code, complex to write for cross platform support (browsers) (DuarTech Solutions, 2021) | Steep learning curve, slower to compile than other languages (Tawfik, 2024) |
| Multi-paradigm? | Yes (Loukides, 2020) | Yes (Loukides, 2020) | Yes (Loukides, 2020) | Yes (Loukides, 2020) |
| Average Salary | $59,454 (Stack overflow developer survey 2021 Top Paying) | $54,049 (Stack overflow developer survey 2021 Top Paying) | $54,049 (Stack overflow developer survey 2021 Top Paying) | $77,530 (Stack overflow developer survey 2021 Top Paying) |
| Web Frameworks | Flask, Django (WebFrameworks - Python Wiki) | Crow, Oat++ (C++ Web Application Framework Libraries) | React, Angular (*100+ javascript frameworks for web developers*) | Actix Web, Rocket (Rust web frameworks: Development in 2023) |

## Selection

The preferred language in this use case will be Python coupled with the Flask web framework. Python was chosen due to its growing popularity and its extreme ease of use. The syntax is very easy to understand and is very close to the English language. It can also handle complex tasks with much less boiler plate code.

I was curious to try C++, but I feel its syntax while close to Java, can be very complex and has a strange use of pointers. JavaScript will not be used solely but I could see JavaScript being used as a companion along with this web development. While Rust seemed interesting, I feel it was too new to take a chance on designing a website application.

Referring to (Saini, 2024), I chose Python as I wanted to use the Flask web framework. This framework is very easy to use and appears to be easily scalable. I also feel this is a very modern way of building a web application as it uses REST style API notation within your main controller app.py. My first iteration will be a Monolithically designed prototype. From this prototype I will refine based on the user requirements and even explore the possibility of a Micro-Service design if time permits. Flask coupled with Python’s MySQL connection framework will allow me to design a basic ticketing system to meet the suggested requirements.

Python flask is the right choice for this project as Python code is a very easy to read and understand out of all the other programming languages compared. This way I can just worry about designing my application and not be burdened with learning complex syntax like the other programming languages.

# Conceptual Design

Just Ticket will be designed with emphasis on design patterns. We will store all our DBConnections into a single class file and practice the singleton pattern. Just Ticket will then use subject specific Data Access Object layer packages, like UserDAO and TicketDAO for CRUD related operations. The heart of our system will be the main app.py and will serve as our controller that will decide on which view is to be displayed to the end user. This will control and record the flow of input and outputs to the persistent Database storage diagramed in the ERD below. We will practice the Model-View-Controller design pattern and the views will consist of html pages that can communicate with the app.py controller that can then query and return the necessary Model information for end user display.

The website portion will be a homepage where they can click to a login page, for now new users will need to be added by an IT admin so there isn’t a register feature. From the login page the user will enter their credentials and from there the main controller will determine based on their role, what view is to be displayed. If the user is an agent, the user be given an Agent dashboard showing all tickets. If they are a customer they will be shown a list of tickets that have been created by them, not all tickets.

From the agent dashboard the agent will be able to review current tickets and provide status updates on tickets. The customer dashboard will have similar features where the customer can provide updates as well, but they will be limited in what tickets they can see. From each list of tickets, the ticket number will be a hyperlink. Upon clicking the user is directed to a ticket detail page. The ticket detail page will be a single view of just one ticket number and will also contain ticket updates in the form of comments, each with a timestamp to record historical activity.

## Package Diagram

A screenshot of a computer program

Description automatically generated

## Class Diagram

**A diagram of a computer

Description automatically generated**

# Entity-Relationship Diagram

A screenshot of a computer program

Description automatically generated

# UI Prototype

**Index.html**

**A black and white rectangular object

Description automatically generated**

**Login.html**

**A screenshot of a computer

Description automatically generated**

**Userdashboard.html**

**A screenshot of a computer

Description automatically generated**

**Createticket.html**

**A screenshot of a computer

Description automatically generated**

**Ticketdetail.html**

**A screenshot of a computer

Description automatically generated**

**References**

*Python introduction*. Introduction to Python. (n.d.). https://www.w3schools.com/python/python\_intro.asp

C++ introduction. (n.d.). https://www.w3schools.com/cpp/cpp\_intro.asp

*C++ Web Application Framework Libraries*. LibHunt. (n.d.). https://cpp.libhunt.com/libs/web-application-framework

*OVH Cloud VPS*. OVHcloud. (n.d.). https://www.ovhcloud.com/en-ca/vps/

Gavriluk, V. (2020, October 22). *What is rust and why you should use it?*. Arounda. https://arounda.agency/blog/what-is-rust-and-why-you-should-use-it

*100+ javascript frameworks for web developers*. CSS Author. (2024, February 29). https://cssauthor.com/javascript-frameworks/

Mittal, C. (2023, March 29). *10 best use cases of rust programming language in 2023*. Medium. https://medium.com/@chetanmittaldev/10-best-use-cases-of-rust-programming-language-in-2023-def4e2081e44

Loukides, M. (2020, November 10). *Multi-paradigm languages*. O’Reilly Media. https://www.oreilly.com/radar/multi-paradigm-languages/

*Rust web frameworks: Development in 2023*. Yalantis. (2024, March 6). https://yalantis.com/blog/rust-web-frameworks/

*What is JavaScript used for?*. BairesDev. (2022, September 5). https://www.bairesdev.com/blog/what-is-javascript-used-for/

WebFrameworks - Python Wiki. (n.d.). https://wiki.python.org/moin/WebFrameworks

Saini, A. (2024, February 7). *Understanding flask framework: Installation, features & expert insights*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/10/flask-python/#:~:text=It%20offers%20a%20straightforward%20and,apps%20to%20meet%20their%20needs.,

Solutions, D. (2021, November 3). *Advantages and disadvantages of the most used programming languages in 2021*. LinkedIn. https://www.linkedin.com/pulse/advantages-disadvantages-most-used-programming-languages-

*Stack overflow developer survey 2021 Most Popular*. Stack Overflow. (n.d.-a). https://survey.stackoverflow.co/2021#section-most-popular-technologies-programming-scripting-and-markup-languages

*Stack overflow developer survey 2021 Top Paying*. Stack Overflow. (n.d.-b). https://survey.stackoverflow.co/2021#top-paying-technologies-programming-scripting-and-markup-languages

Tawfik, M. (2024, February 19). *Rust language: Pros, cons, and learning guide*. Medium. https://medium.com/@apicraft/rust-language-pros-cons-and-learning-guide-594e8c9e2b7c#:~:text=Cons%20of%20Rust&text=Lesser%20Library%20Support%3A%20While%20growing,compared%20to%20more%20established%20languages.