

Paradox  
Studios

**Prepared By :**

Talia Dhanampal  
Pranhav Maistry  
Trinity Jayd Tiratram,  
Yusra Tareen Adnan  
Rebekah-Leigh Naidoo,  
Saffiyah Mansoor  
Mason Buist  
Jessica Leigh De Sousa Caldeira

# CONTENTS

<b><u>Introduction to project</u></b>	4
Our Team and Client	4
Project Outline	5
Progression Discussion	6
<b><u>Work agreement</u></b>	7
Team work agreement and roles	11
Definition of ready (DoR)	12
Definition of Done (DoD)	15
Roadmap	20
<b><u>Requirements</u></b>	23
User Roles	23
User Stories	26
User Experience Journey Map	27
<b><u>Progression Discussion</u></b>	29
<b><u>Project contents/documentation UML</u></b>	30
<b><u>Project contents/documentation Burn Downs</u></b>	31
<b><u>Non-functional requirements</u></b>	32

<b><u>Data Schema Documentation</u></b>	34
<b><u>Architecture artifacts</u></b>	35
Architecture Patterns	35
<b><u>Security</u></b>	37
<b><u>DevOps</u></b>	39
<b><u>Running costs</u></b>	42
<b><u>Change Management</u></b>	44



# **Introduction to project**

## **Our Team & Client**

### **Team Leader**

Talia Dhanampal  
[ST10083706@vcconnect.edu.za](mailto:ST10083706@vcconnect.edu.za)

### **Back-end Leader**

Pranhav Maistry  
[ST10083716@vcconnect.edu.za](mailto:ST10083716@vcconnect.edu.za)

### **Main Back-end Members**

Yusra Tareen Adnan  
[ST10083751@vcconnect.edu.za](mailto:ST10083751@vcconnect.edu.za)

Jessica Leigh De Sousa Caldeira  
[ST10083342@vcconnect.edu.za](mailto:ST10083342@vcconnect.edu.za)

Mason Buist  
[ST10083160@vcconnect.edu.za](mailto:ST10083160@vcconnect.edu.za)

### **Our Client**

Durban Institute of Technology  
(DUT)

### **Front-end Leader**

Trinity Jayd Tiratram  
[ST10083735@vcconnect.edu.za](mailto:ST10083735@vcconnect.edu.za)

### **Main Front-end Members**

Trinity Jayd Tiratram  
[ST10083735@vcconnect.edu.za](mailto:ST10083735@vcconnect.edu.za)

Saffiyah Mansoor  
[ST10083932@vcconnect.edu.za](mailto:ST10083932@vcconnect.edu.za)

Rebekah-Leigh Naidoo  
[ST10083773@vcconnect.edu.za](mailto:ST10083773@vcconnect.edu.za)

# Introduction:

The DUT App Factory project is a ticket assist system designed to serve the students and staff of the university DUT (Durban University of Technology). This project is intended to facilitate university operations and services by providing a ticket-logging platform for creating, managing and resolving tickets or requests.

## Background:

In the initial client meeting, the support team at DUT has requested a web app that:

- Handles queries that are related to their form system that allows students and teachers to submit queries.
- Utilise Talk - a live chat feature that assists with problems
- Handles email queries by responding to each email on the support email.
- This is time-consuming because there are a lot of emails so automated email scripts are requested.
- Automate the email process by creating tickets for each email received.
- Send out an automatic response stating that the query will be handled within a specific period of time, ie: 24hrs. This is the Escalation feature to notify staff members/ team members once 24 hours are up.
- Team members should be able to assign tickets to one another
- Have the ability to open up a ticket for bugs to be notified to their developers
- Create email templates to have consistency in customer communication and reduce the time spent writing emails

# Introduction:

## Feedback and Analytics:

- The support team requires a rating system from users who use the system such as how their engagement was, how the query was resolved, and general experience.
- Generate a report on how many queries have been opened, closed, and how many queries each team member has handled.

The key objective of this system is to allow users to create, submit, and track tickets or requests related to various university services and issues. It aims to improve user engagement and communication by providing an efficient channel for students and staff to interact with university departments or support teams. The system helps in streamlining and automating ticket routing and resolution processes, potentially reducing response times and enhancing overall operational efficiency. It includes features for data reporting to help the university assess service quality and make data-driven decisions to enhance the student and staff experience.

The system is designed to be accessible through both web or mobile interfaces, ensuring that users can easily submit and track their tickets from various devices.

# Work Agreement:

The work agreement conditions:



## **WORK INTEGRATED LEARNING (XBCAD7319)** **PARADOX STUDIOS TEAM CONTRACT**

### **1. Purpose**

The purpose of this Contract is to establish the expectations, responsibilities, and boundaries of each member of Paradox Visions to ensure effective collaboration and achievement of the Team's goals.

### **2. Roles and Responsibilities**

Each member of the team shall have the following roles and responsibilities:

2.1. Attend all Team meetings and actively participate in discussions and decision-making. If a team member is unable to attend, they will notify the team leader with valid reasoning.

2.2. Complete assigned tasks and deliverables on time of the given deadline and to the best of their ability.

2.3. If a team member is unable to complete assigned tasks, they will notify the team leader with valid reasoning. A reasonable number of chances will be given to re-submit assigned tasks before the tasks will be allocated to other team

members who are able and available to do it.

2.4. If a team member is unable to complete assigned tasks without valid reasoning, credit will not be given to said team member and instead credit will be given to team members who complete the task.

2.5. Communicate regularly and openly with the Team regarding progress, concerns, and ideas.

2.6. Respect the opinions and contributions of other Team members and work collaboratively towards the Team's goals.

2.7. Adhere to any applicable laws, regulations, and ethical standards in the performance of their duties.

2.8. Put any personal bias of fellow team members aside when working on tasks and participating in decision-making.

# Work Agreement:

The work agreement conditions:

## **3. Communication**

Communication will occur through the following channels:

3.1. Scheduled team meetings and check-ins

3.2. Email or other messaging platforms

3.3. Other channels as deemed necessary by the Team

The Team shall maintain regular once-a-week meetings.

3.4. Whether the meetings will be online or face-to-face will be communicated prior to the meeting.

3.5. Online meetings will be on a platform agreed upon by the team.

3.6. If meetings are unable to be had the team leader will decide to either reschedule on an appropriate date or send an email detailing the topics that need to be acknowledged and discussed.

## **4. Conflict Resolution**

In the event of conflicts or disagreements among Team members, the Team shall follow the following conflict resolution process:

4.1. Discuss the issue openly and honestly with the other members involved.

4.2. Seek input and advice from the team leader, other Team members, and/or stakeholders, if necessary.

4.3. Work collaboratively to find a mutually agreeable solution or agree to vote amongst team members to come to a unanimous decision.

4.4. If the issue cannot be resolved internally, seek assistance from a neutral third party, if necessary.

## **6. Termination**

If a member wishes to terminate their membership in the Team, they shall provide written notice to the other members at least five days prior to their intended departure date. The departing member shall complete any outstanding tasks and deliverables prior to their departure.

## **8. Entire Agreement**

This Contract constitutes the entire agreement between the parties and supersedes all prior negotiations, understandings, and agreements between the parties.

IN WITNESS WHEREOF, the parties have executed this Contract as of the date first above written.

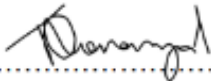
This contract will be applicable from the date signed.



# Work Agreement:

The work agreement conditions:

Thus done and signed at Varsity College(Durban North) on this .....23..... day of  
.....May..... 2023.



ST10083706

Talia Dhanampal

Thus done and signed at Varsity College(Durban North) on this .....23..... day of  
.....May..... 2023.



ST10083773

Rebekah-Leigh Naidoo

Thus done and signed at Varsity College(Durban North) on this .....24..... day of  
.....May..... 2023.



ST10083751

Yusra Tareen Adnan

Thus done and signed at Varsity College(Durban North) on this .....23..... day of  
.....May..... 2023.




ST10083735

Trinity Jayd Tiratram

# Work Agreement:

The work agreement conditions:

Thus done and signed at Varsity College(Durban North) on this ...23... day of  
..... May ..... 2023.




ST10083932  
Saffiyah Mansoor

Thus done and signed at Varsity College(Durban North) on this ...23... day of  
..... May ..... 2023.



ST10083342  
Jessica Leigh De Sousa Caldeira

Thus done and signed at Varsity College(Durban North) on this ...23... day of  
..... May ..... 2023.



ST10083716  
Pranhav Maistry

Thus done and signed at ..... Home ..... on this ...3... day of  
..... July ..... 2023.



ST10083160  
Mason Buist

# Work Agreement:

## Team Responsibilities:

Talia Dhanampal ST10083706@vcconnect.edu.za  
Team Leader

Pranhav Maistry ST10083716@vcconnect.edu.za  
Development Back-end Leader

Trinity Jayd Tiratram  
ST10083735@vcconnect.edu.za  
Development Front-end Leader

## Requirements and Testing:

Yusra Tareen Adnan ST10083751@vcconnect.edu.za

Trinity Jayd Tiratram

ST10083735@vcconnect.edu.za

Saffiyah Mansoor ST10083932@vcconnect.edu.za

Jessica Leigh De Sousa Caldeira

ST10083342@vcconnect.edu.za

Mason Buist ST10083160@vcconnect.edu.za

## Architecture and Design

Trinity Jayd Tiratram

ST10083735@vcconnect.edu.za

Rebekah-Leigh Naidoo

ST10083773@vcconnect.edu.za

# Work Agreement:

Definition of Ready:

Sprint 1: From 4 August to 8 September

User Story:

I, as a user, should be able to login/signup to the web application so that the support team can identify me

Acceptance Criteria:

Allows users to login to their user account using domain login.

This will allow the admin to determine a user's status.

User Story:

I, as a user, should be able to submit/log a ticket to the support team so that they can attend to my issue.

Acceptance Criteria:

Creates a ticket with details: User ID, topic/subject of query, user query, and supporting documents.

User Story:

I, as a user, should be able to choose the category of subject for my ticket when creating a ticket.

Acceptance Criteria:

When the user chooses their category the support team will know what department the query is concerning.

Sprint 2: 9 September - 1 October

User Story:

I, as a support member, should be able to view a ticket and its history.

Acceptance Criteria:

The ticket view will display the ticket details, the priority rating, the date that the ticket was made and how much work was put into resolving the query.

# Work Agreement:

## User Story:

As the system, it should be able to close a ticket when it is completed so that it is removed from the pending tickets.

## Acceptance Criteria:

When the support team member changes a ticket's status to resolved it will

The system should be able to assign priority to user queries.

The system should be able to detect keywords and assign a priority rating for admin to differentiate between low, medium and high-priority requests.

## User Story:

As the system, it should be able to send automated responses of acknowledgement to student support queries so that support members can save time.

## Acceptance Criteria:

The responses will be sent via email.

It will be an automated template that will be sent.

The system will also send the user ticket number and confirmation of their priority rating.

## User Story:

I, as a lead support member, should be able to assign tickets to other support members so that I am able to delegate tasks and track what each member is currently working on.

## Acceptance Criteria:

Given that tickets need to be worked on when I assign a ticket to a member, they are notified and are assigned the ticket on the system.

# Work Agreement:

## User Story:

I, as a support team member should be able to change a ticket completion status from: posted (red), in progress (yellow) and resolved (green).

## User Story:

I, as a user, should be able to view a ticket and its history.

## Acceptance Criteria:

The user will only be able to view, in a "Tickets posted" section, the tickets they've posted and their related information and whether it has been resolved or not.

## User Story:

I, as a support team member, should be able to reply with predefined template emails so that an increase in speed and productivity can be achieved.

## Acceptance Criteria:

Templates should be saved on the system and when replying to tickets they can copy and paste the templates.

As the system, it should be able to identify the queries according to the priority rating when it hasn't been resolved within the respective times.

## User Story:

As the system, it should be able to log the query status for data analysis.

## User Story:

As the system, it should be able to generate a monthly report which details the total queries, received and respond to so that data is easily available when needed.

## Acceptance Criteria:

The report should be automatically generated on the system and ready for the user to download.

## User Story:

I, as a user, should be allowed to rate the experience of the support for customer service reasons as well as be able to comment.

## Acceptance Criteria:

At the end of the experience, a pop-up will appear and ask a user to rate them.

# Work Agreement:

Definition of Done:

Sprint 1:

Definition of done:

User Story ID	User Story	Unit Tests	Code Peer Reviewed	Acceptance Criteria	Functional Tests	Non-Functional requirements
1	I, as a user, should be able to login/sign up to the web application so that the support team can identify me.	Done	Done	Met	Passed	Met
2	I, as a user, should be able to submit/log a ticket to the support team so that they can attend to my issue.	Done	Done	Met	Passed	Met
3	I, as a user, should be able to choose the category of subject	Done	Done	Met	Passed	Met

# Work Agreement:

Definition of Done:

Sprint 1:

	for my ticket when creating a ticket.					
4	I, as a support member, should be able to view a ticket and its history.	Done	Done	Met	Passed	Met
5	As the system, it should be able to close a ticket when it is completed so that it is removed from the pending tickets.	Done	Done	Met	Passed	Met



# Work Agreement:

Definition of Done:


Sprint 2:

6	The system should be able to assign priority to user queries.	Done	Done	Met	Passed	Met
7	As the system, it should be able to send automated responses of acknowledgement to student support queries so that support members can save time.	Done	Done	Met	Passed	Met
8	I, as a lead support member, should be able to assign tickets to other support members so that I am able to delegate tasks and track what each member is currently working on.	Done	Done	Met	Passed	Met
9	I, as a support team member should be able to change a ticket completion status from: posted (red), in progress (yellow) and resolved (green)	Done	Done	Met	Passed	Met
10	I, as a user, should be able to view a ticket and its history. - This is almost done already just need to make the view a bit more intuitive	Done	Done	Met	Passed	Met
11	I, as a support team member, should be able to reply with predefined template emails so that an increase in speed and productivity can be	Done	Done	Met	Passed	Met

# Work Agreement:

Definition of Done:

Sprint 2:

12	As the system, it should be able to identify the queries according to the priority rating when it hasn't been resolved within the respective times.	Done	Done	Met	Passed	Met
13	As the system, it should be able to log the query status for data analysis.	Done	Done 	Met	Passed	Met
14	As the system, it should be able to generate a monthly report which details the total queries, received and respond to so that data is easily available when needed.	Done	Done	Met	Passed	Met
15	I, as a user, should be allowed to rate the experience of the support for customer service reasons as well as be able to comment.	Not Done	Not Done	Not Met	Not Passed	Not Met
16	I , as a user should be able to attach documents and screenshots to a query or response so that i am able to provide more information for my query.					
17	Lead should be able to view all members devs and what tickets they are currently working on					

# Work Agreement:

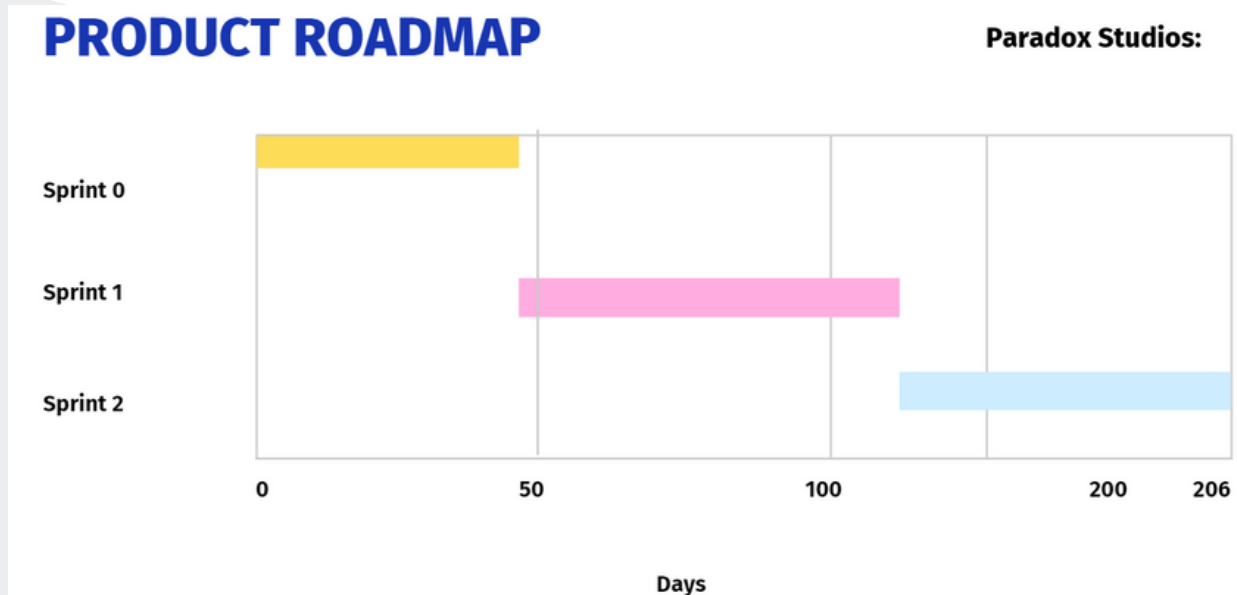
Definition of Done:

Sprint 2:

18	I, as the user, should be able to view all documents submitted and be able to download them,					
19	Dev should be able to add notes to a ticket and the notes be stored to that ticket					

# Work Agreement:

Roadmap:



Time commitments:

Daily Scrum: 15 minutes on weekdays (find in meeting minutes)

Backlog Grooming: 1 Hour per day, over 5 days in sprint 0. 5 Hours total

Sprint Planning: 2 Hours in sprint 0.

Sprint Review: 1 Hour in sprint 0.

Sprint Retrospective: 1 Hour in sprint 0.

Task Work: 156 days for 4 hours. 624 hours Total

# Work Agreement:

## **Sprint 0: Plan**

Agile Life Cycle phase: Conceptualisation and Inception

Sprint Goal: Meet with the client, develop user stories and requirements, as well as a first draft of the user journey, define sprints and assign team roles.

Team Availability:

As a developer I need a prioritised backlog of requirements from the client so that I can develop an application that meets their needs.

As a developer, I need to understand how the screens and user roles will work together, so that I can build a UI that meets the client's needs.

## **Sprint 1: Build**

Agile Life Cycle phase: Iteration

Sprint Goal: The user should be able to perform basic functions of the app with minimal bugs.

Team Availability:

As a frontend developer I need to know which framework to use and which standards to follow so that I can develop high quality code.

As a DevOps team member, I need to understand how GitHub actions work so that I can build a pipeline to ensure a high level of quality of the code that is committed.

As a backend developer I need to know which libraries (SDK) versions to use for backend development so that I can ensure my local environment is configured to match the rest of my team.

# Work Agreement:

As a backend developer, I need a comprehensive database schema and data model documentation so that I can efficiently work with the database and create or modify database queries.

As a backend developer, I need access to clear API documentation and endpoints, including expected request and response formats, to ensure effective integration with frontend components.

## **Sprint 2: Build, Deploy and Present**

Agile Life Cycle phase: Release and Maintenance

Sprint Goal: Complete full functionality of web-app and deploy.

Team Availability:

As a frontend developer, I need access to a comprehensive style guide and design specifications so that I can maintain consistency in the user interface and deliver a seamless user experience.

As a DevOps team member, I need to understand the deployment process for our application in different environments (e.g., development, staging, production) to ensure a smooth and reliable release process

As a backend developer, I need to know the data encryption standards and practices used in the application to secure sensitive information and transactions.

As a backend developer, I need information on the data validation and input sanitization procedures to ensure the security and integrity of user inputs.

# Requirements:

User roles:

Students:

Students use the ticket system to submit requests or report issues related to university services. These requests can include IT support, maintenance, library inquiries, administrative issues, or any other service-related concerns.

Students can track the progress and status of their submitted tickets. This feature allows them to stay informed about when their request will be addressed or resolved.

The ticket system may facilitate communication between students and university staff regarding their submitted tickets. This can include clarifications, updates, or additional information requests.

After a ticket is resolved, students may have the option to provide feedback on the quality of service received. This feedback can be valuable for improving university services.

Staff:

Admin/Developers:

# Requirements:

## Staff:

University staff members, including faculty, administrators, and support personnel, use the system to manage and respond to incoming tickets. They can assign, prioritize, and track the progress of tickets within their respective departments.

Staff can use the system to communicate with students, asking for more information or providing updates on ticket status. This feature ensures clear and efficient communication.

Staff members are responsible for resolving the tickets within their areas of expertise or responsibility. This could involve providing solutions, performing maintenance, or offering guidance.

Some staff roles might use the system's data and analytics features to analyze ticket trends, monitor service quality, and make data-driven decisions to improve their department's performance.



# Requirements:

## Admin/Devs:

Administrators, often developers or IT personnel, manage the technical aspects of the ticket system. They ensure the system's availability, security, and functionality.

Admins maintain user accounts, permissions, and roles within the ticket system, ensuring that the right people have appropriate access.

Administrators may customize the system to meet the specific needs of the university. This could include configuring workflows, adding new features, or integrating the system with other university applications.

Admins use the system's reporting tools to monitor system performance, troubleshoot issues, and provide insights into how the system is being used. This data helps in system optimization.


# Requirements:

## User Stories:

+	1	User Story	As a user, I should be able to login/signup to the web application so that the support team can identify me.
	2	User Story	I, as a user, should be able to submit/log a ticket to the support team so that they can attend to my issue.
	3	User Story	I, as a user, should be able to choose the category of subject for my ticket when creating a ticket.
	4	User Story	I, as a support member, should be able to view a ticket and its history.
	5	User Story	As the system, it should be able to close a ticket when it is completed so that it is removed from the pending tickets.
	6	User Story	The system should be able to assign priority to user queries.
	7	User Story	As the system, it should be able to send automated responses of acknowledgement to student support queries so that support members can save time.
	8	User Story	I, as a lead support member, should be able to assign tickets to other support members so that I am able to delegate tasks and track what each member is currently working on.
	9	User Story	I, as a support team member should be able to change a ticket completion status from: posted (red), in progress (yellow) and resolved (green).
	10	User Story	I, as a user, should be able to view a ticket and its history.
	11	User Story	I, as a support team member, should be able to reply with predefined template emails so that an increase in speed and productivity can be achieved.
	12	User Story	As the system, it should be able to identify the queries according to the priority rating when it hasn't been resolved within the respective times.
	13	User Story	As the system, it should be able to log the query status for data analysis.
	14	User Story	As the system, it should be able to generate a monthly report which details the total queries, received and respond to so that data is easily available when needed.
	15	User Story	I, as a user, should be allowed to rate the experience of the support for customer service reasons as well as be able to comment.

# Requirements:

## Journey Map:



LOGOUT YOUR TICKETS LOG A TICKET

Start Date  
yyyy/mm/dd


End Date  
yyyy/mm/dd

Status  
All

Category  
All

Filter

No tickets found



LOGOUT YOUR TICKETS LOG A TICKET

Log a Ticket

Please select a category for your query to continue

IT/Support

LOG

Admissions

LOG

Results


LOG

Student Wellness

LOG

Submissions

LOG



LOGOUT YOUR TICKETS LOG A TICKET

Student Information


Category  
IT/Support

Query

Message

Choose files No file chosen

Submit



LOGOUT YOUR TICKETS ADMIN LOG A TICKET

My Tickets

Search By Ticket ID

Search

Start Date  
yyyy/mm/dd

End Date  
yyyy/mm/dd


Status  
All

Category  
All


Priority  
All

Filter


No tickets found




LOGOUT YOUR TICKETS ADMIN LOG A TICKET



My Tickets



All Tickets



Analytics

Ticket Count by Date

Ticket Status Breakdown

Priority Breakdown

PARADOX STUDIOS

27

# UX Design

## User Experience

Our website showcases our exceptional design skills, showing a thoughtful and intuitive approach that makes the user's experience truly enjoyable. By combining a keen eye for aesthetics with a deep understanding of user behaviour, we've created a digital space that captivates users from the moment they land on our homepage. Navigating through our website is simple, thanks to the well-organized layout, logical menu structure, and seamless transitions between pages. Every element, from the colour palette to the interactive features, has been meticulously crafted to enhance user engagement. Our commitment to delivering an enjoyable experience shines through every click and interaction, making our website a delightful destination for all who visit.

Our keen attention to user experience has led to thoughtful design choices that resulted in a seamless online platform. We did this by prioritizing intuitive navigation, ensuring that users can effortlessly find the information they seek, enhancing their overall experience. Our commitment to user-friendliness not only showcases our dedication to providing a positive online journey but also demonstrates our ability to create an accessible and engaging digital space. Our website is a testament to our exceptional design skills, making it a pleasure for users to interact with.

# Progression Discussion

## MEETING MINUTES

### AUGUST

08	12:00 - 12:30 pm	(Team Meeting)
15	13:00 - 14:00 pm	(Mentor Meeting)
16 - Present	15:00 - 15:15 pm	(Team Meeting)

### SEPTEMBER

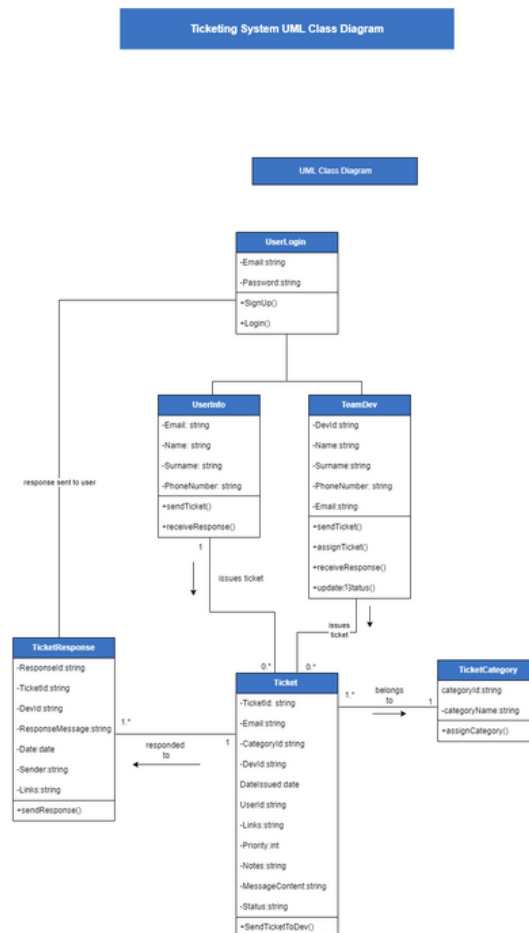
08	10:30 am	(Mentor Meeting)
12	14:30 pm	(Mentor Meeting)
21	16:00 pm	(Mentor Meeting)

### OCTOBER

25	14:00 pm	(Team Meeting)
27	13:00 pm	(Team Meeting)
27	14:00 pm	(Mentor Meeting)
30	15:00 pm	(Team Meeting)

# Project contents/ documentation:

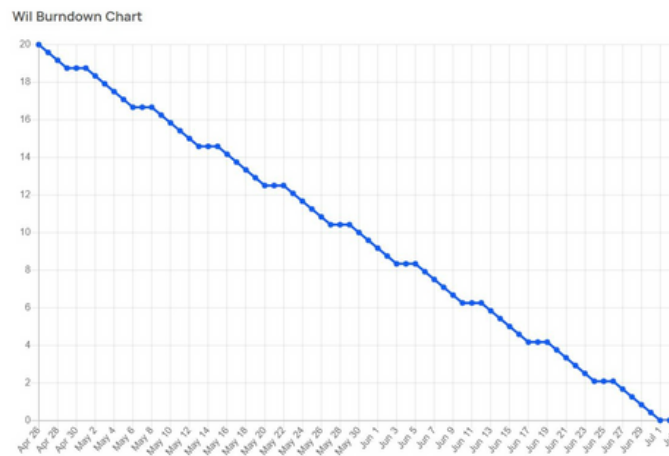
## UML Diagram



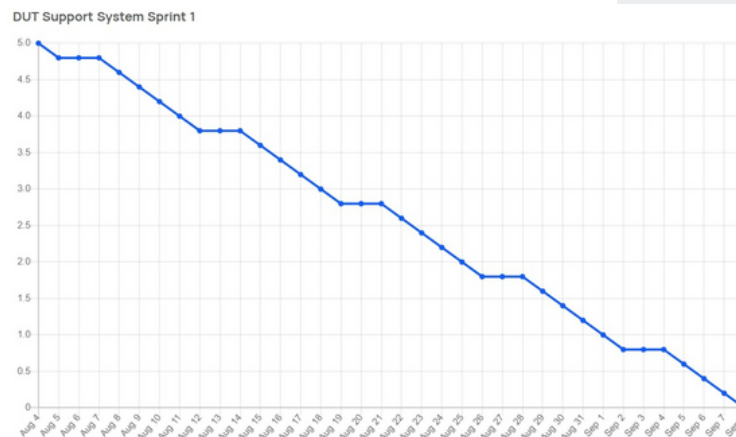
# Project contents/ documentation:

## Burn Down Charts

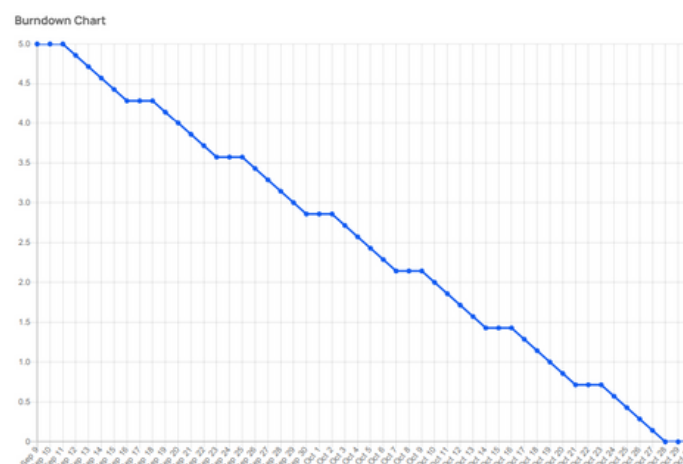
### Sprint 0:



### Sprint 1:



### Sprint 2:



# **Non-functional requirements**

1. **Performance:** The system should respond within 15 seconds for user interactions, and it should be able to support up to 350 concurrent users efficiently. This ensures that the system remains responsive and usable even under load.
2. **Scalability:** The system should exhibit a high level of scalability, meaning it can effectively handle increased loads and user demands as the need arises. This ensures that the system can grow with the user base.
3. **Security:** The system must exhibit a high level of security to protect user data and prevent unauthorized access. This is crucial for maintaining the confidentiality and integrity of sensitive information.
4. **Reliability:** The system should be moderately reliable, indicating that it should be available for use with acceptable downtime and should have a regular backup strategy in place to minimize data loss.
5. **Usability:** The system should be very easy to use, ensuring an intuitive and user-friendly interface. It should also be fairly documented to assist users in understanding its features and functionality.
6. **Maintainability:** The system should have moderate maintainability, meaning it should be reasonably easy to maintain and update, allowing for the addition of new features and improvements.
7. **Portability:** The system should be very easily portable to different platforms, ensuring it can run on a variety of hardware and software environments without significant modification.
8. **Localization:** The system should have moderate localization capabilities, allowing for adaptation to different languages and cultures. This ensures it can effectively cater to a diverse user base.



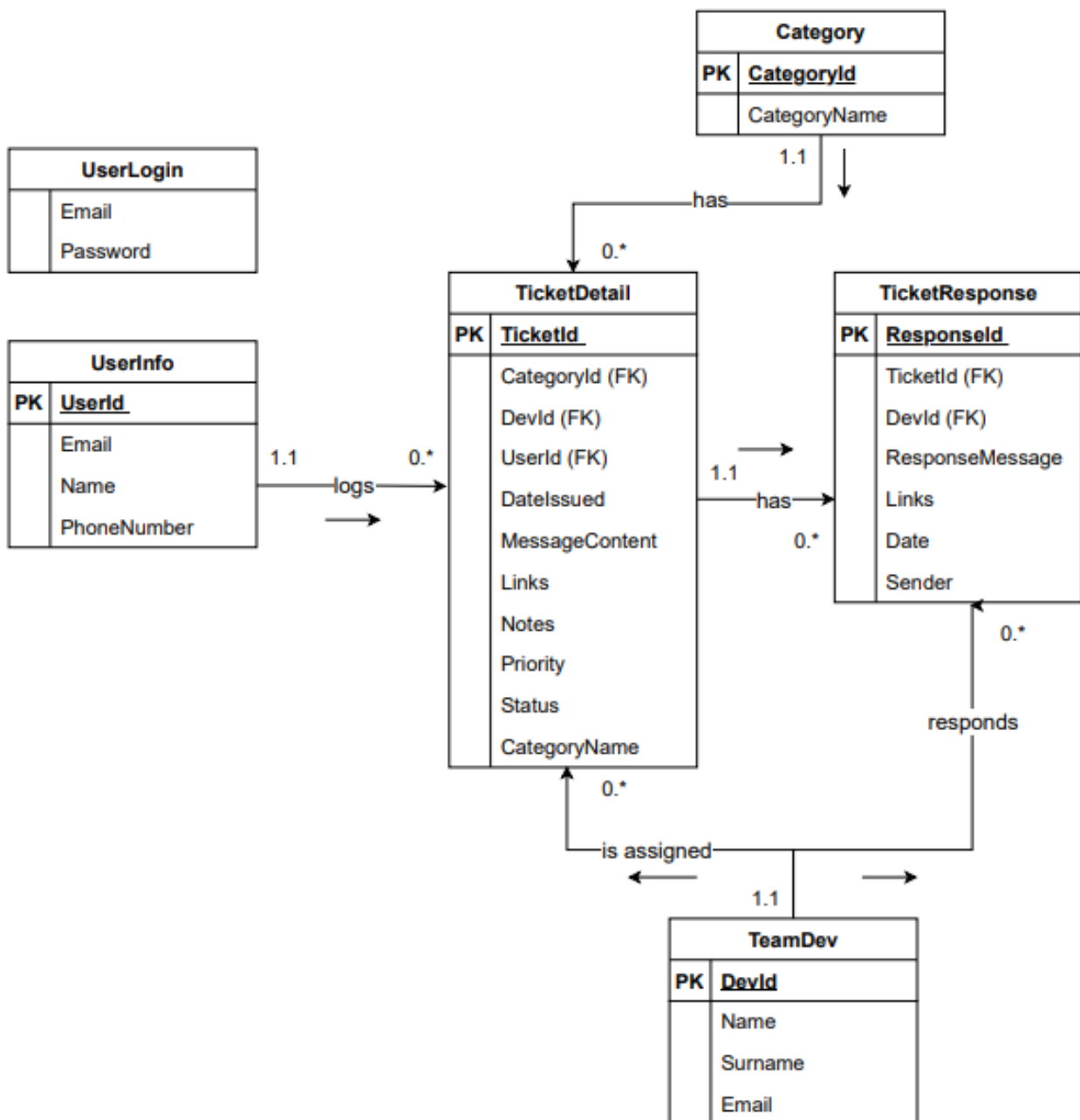
# **Non-functional requirements**

9. Accessibility: The system should have moderate accessibility, meaning it should be designed to be usable by individuals with disabilities, following accessibility standards and guidelines to enhance inclusivity.

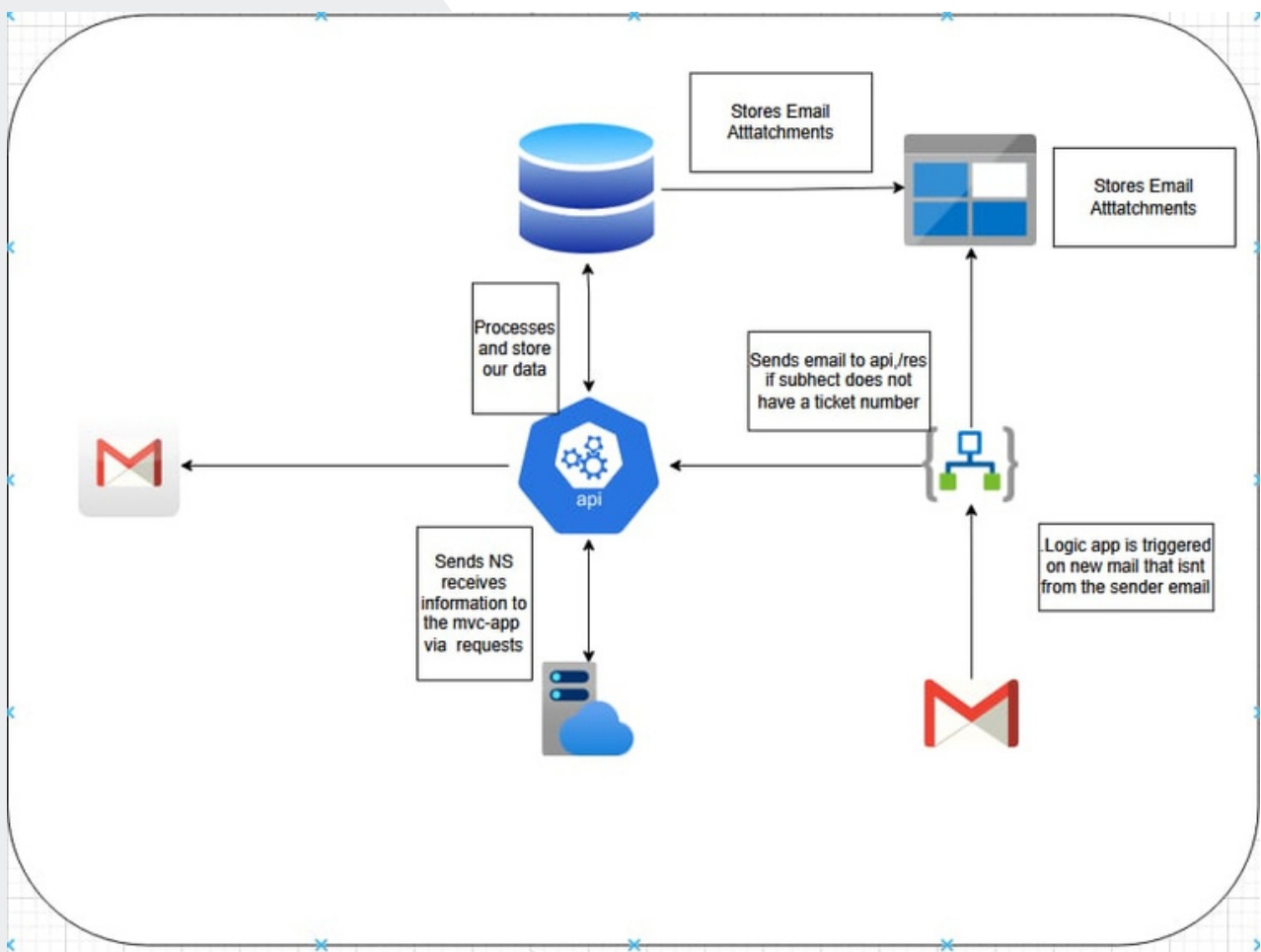
# Data Schema

# Documentation

DUT App Factory V3.0



# Architecture artifacts



# Architecture artifacts

We have implemented Microservice Architecture which offers:

## **Scalability and Flexibility**

The choice of a microservice architecture reflects a deliberate effort to design the system for scalability and flexibility. Instead of building a monolithic application, which can become unwieldy and less adaptable as it grows, the system is structured as a collection of independent microservices. Each microservice focuses on a specific aspect of the application's functionality.

## **Complexity**

Microservices break down the complexity of the application into manageable, self-contained units. Each microservice can be developed, deployed, and scaled independently. This decomposition allows development teams to work on different services concurrently.

## **Responsibility and Isolation**

Each microservice is responsible for a specific set of tasks or functionalities. This isolation ensures that changes or issues in one microservice do not adversely affect the entire system. It promotes a high degree of fault tolerance and resilience.

# Security

Security in the application is implemented through several measures. Passwords are securely stored using the bcrypt hashing algorithm, which adds an extra layer of protection to user credentials. The application is hosted on Azure, utilizing HTTPS for secure data transmission. Azure data storage security features are also employed to protect sensitive information from unauthorized access.

**Bcrypt Password Hashing:** Bcrypt is a secure password hashing algorithm. It adds an extra layer of protection to user credentials by securely storing and managing passwords. When a user creates an account or updates their password, the system uses bcrypt to hash the password. The hashed password is stored in the database, not the plain text password. This significantly enhances security because even if the database is compromised, the actual passwords remain hidden, as bcrypt hashes are computationally expensive to reverse-engineer.

**Azure Hosting and HTTPS:** The application is hosted on the Azure platform, a trusted and reputable cloud service provider. Azure offers a range of security features, including robust firewall configurations, intrusion detection systems, and regular security updates. By hosting the application on Azure, it benefits from these built-in security measures.

**HTTPS (Hypertext Transfer Protocol Secure):** HTTPS is employed for secure data transmission. All data exchanged between the application and its users is encrypted during transit. It provides end-to-end encryption, ensuring that data is secure during communication. It's especially crucial for protecting sensitive information, such as user credentials and personal data, from eavesdropping and man-in-the-middle attacks.

# Security.


















**Azure Data Storage Security:** Azure data storage security features are implemented to safeguard sensitive information. Azure provides a suite of security tools, including access controls, encryption at rest, and threat detection. Data is protected both in transit and at rest, preventing unauthorized access and data breaches.




**Access Control and Authorization:** To further enhance security, the application likely employs access control and authorization mechanisms. Users are granted access only to the data and functionalities they are authorized to use. This minimizes the risk of unauthorized access to sensitive data or actions within the application.

**Regular Security Updates:** A proactive approach to security includes staying up-to-date with security patches and updates. Regular updates to the application's software, frameworks, and libraries help to address known vulnerabilities and keep the system secure.


















**Security Audits and Testing:** Periodic security audits and testing, such as penetration testing and vulnerability assessments, may be conducted to identify and rectify security weaknesses. This ensures that potential security risks are addressed in a timely manner.

# DevOps

 	Order	Work Item Type	Title	State	Effort	Busin...	Value Area	Tags
	1	Epic	>  User Authentication and Account Management	... ● New			Business	
	2	Epic	>  Support Ticket Management	● New			Business	
	3	Epic	>  Ticket Categorization and Routing	● New			Business	
	4	Epic	>  Ticket Viewing and Management	● New			Business	
	5	Epic	>  Ticket Closure and Archiving Automation	● New			Business	
	6	Epic	>  Priority Management and Optimization	● New			Business	
	7	Epic	>  Automated Acknowledgment System	● New			Business	
	8	Epic	>  Ticket Assignment and Workload Management	● New			Business	
	9	Epic	>  Ticket Status and Progress Management	● New			Business	
	10	Epic	>  Predefined Email Templates and Productivity Enhancement	● New			Business	
	11	Epic	>  Priority-Based Query Identification and Escalation	● New			Business	
	12	Epic	>  Query Status Data Logging and Analysis	● New			Business	
	13	Epic	>  Monthly Query Reporting and Data Availability	● New			Business	
	14	Epic	>  Customer Feedback and Service Improvement	● New			Business	

 DUT App Factory Team   

**Backlog** Analytics  New Work Item  View as Board  Column Options ...

 	Order	Work Item Type	Title	State	Effort	Busin...	Value Area
	1	Feature	>  User Authentication	... ● New			Business
	2	Feature	>  Ticket Submission and Management	● New			Business
	3	Feature	>  Ticket Categorization	● New			Business
	4	Feature	>  Ticket Viewing and History	● New			Business
	5	Feature	>  Ticket Closure and Archiving	● New			Business
	6	Feature	>  Priority Assignment for User Queries	● New			Business
	7	Feature	>  Automated Acknowledgment Responses	● New			Business
	8	Feature	>  Ticket Assignment and Delegation	● New			Business
	9	Feature	>  Ticket Status Management	● New			Business
	10	Feature	>  Predefined Email Templates	● New			Business
	11	Feature	>  Priority-Based Query Identification	● New			Business
	12	Feature	>  Query Status Logging	● New			Business
	13	Feature	>  Monthly Query Reporting	● New			Business
	14	Feature	>  Customer Feedback and Rating	● New			Business

# DevOps

DUT App Factory Team

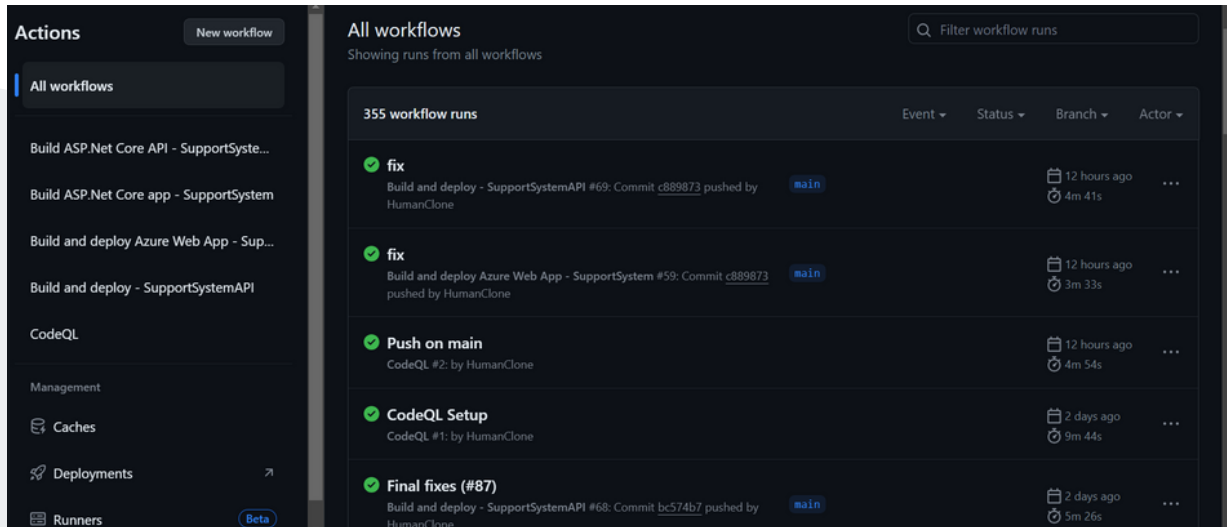
Backlog Analytics + New Work Item View as Board Column Options

Stories

	Order	Work Item Type	Title
+	1	User Story	I, as a user, should be able to login/signup to the web application so that the support team can identify me.
	2	User Story	I, as a user, should be able to submit/log a ticket to the support team so that they can attend to my issue.
	3	User Story	I, as a user, should be able to choose the category of subject for my ticket when creating a ticket.
	4	User Story	I, as a support member, should be able to view a ticket and its history.
	5	User Story	As the system, it should be able to close a ticket when it is completed so that it is removed from the pending tickets.
	6	User Story	The system should be able to assign priority to user queries.
	7	User Story	As the system, it should be able to send automated responses of acknowledgement to student support queries so that support members can save time.
	8	User Story	I, as a lead support member, should be able to assign tickets to other support members so that I am able to delegate tasks and track what each member is currently working on.
	9	User Story	I, as a support team member should be able to change a ticket completion status from: posted (red), in progress (yellow) and resolved (green).
	10	User Story	I, as a user, should be able to view a ticket and its history.
	11	User Story	I, as a support team member, should be able to reply with predefined template emails so that an increase in speed and productivity can be achieved.
	12	User Story	As the system, it should be able to identify the queries according to the priority rating when it hasn't been resolved within the respective times.
	13	User Story	As the system, it should be able to log the query status for data analysis.
	14	User Story	As the system, it should be able to generate a monthly report which details the total queries, received and respond to so that data is easily available when needed.
	15	User Story	I, as a user, should be allowed to rate the experience of the support for customer service reasons as well as be able to comment.



# Pipeline



The screenshot displays the GitHub Actions 'All workflows' page. On the left, a sidebar lists various workflow categories: 'All workflows' (selected), 'Build ASP.Net Core API - SupportSystem...', 'Build ASP.Net Core app - SupportSystem', 'Build and deploy Azure Web App - Sup...', 'Build and deploy - SupportSystemAPI', 'CodeQL', 'Management', 'Caches', 'Deployments', and 'Runners' (marked as Beta). The main panel, titled 'All workflows', shows a list of 355 workflow runs. The runs are filtered by 'main' branch and include details such as the event type (e.g., 'fix', 'Push on main', 'CodeQL Setup', 'Final fixes (#87)'), the commit hash, the actor (HumanClone), and the time since the run completed (e.g., '12 hours ago', '2 days ago').

Event	Status	Branch	Actor
fix	Completed	main	HumanClone
fix	Completed	main	HumanClone
Push on main	Completed	main	HumanClone
CodeQL Setup	Completed	main	HumanClone
Final fixes (#87)	Completed	main	HumanClone

# Running costs

## Cost Benefit:

- **Cost:** The estimated cost of the system is \$625 in full deployment. During development, the costs are lower, around \$100. These costs likely cover expenses associated with cloud hosting, secure data storage, scalability, development, and maintenance.
- **Benefits:** The system offers several benefits, including enhanced security through bcrypt encryption and Azure hosting with HTTPS. The use of microservices architecture allows for scalability and flexibility. Advanced techniques like workflows and CI/CD processes improve development efficiency. However, it's important to note that unit tests were not mentioned, which could impact the overall quality of the system.

## Cost:

**Full Deployment (\$625):** The estimated cost of \$625 for full deployment encompasses various expenses required to bring the system into a production-ready state. These costs are associated with different aspects of the project, including:

**Cloud Hosting:** A significant portion of the cost goes toward cloud hosting services. Hosting on cloud platforms like Azure provides scalability, reliability, and security but comes with associated fees, including server usage, storage, and data transfer costs.

**Secure Data Storage:** Ensuring secure data storage is paramount, and this includes costs related to setting up and maintaining databases with robust security features. These expenses may involve encryption, access control, and redundancy to protect user data.

**Scalability:** Building a system that can scale as user demand increases often requires additional investment in infrastructure, load balancing, and redundancy to ensure high availability.

# Running costs

Development: Costs during development are lower (around \$100), as they primarily cover the expenses of software development, including developer salaries, software licenses, and development tools.

Maintenance: Ongoing maintenance and updates to the system may require additional costs for bug fixes, feature enhancements, and addressing security vulnerabilities.

Predicted costs of running solution:

The following needs to be taken into consideration:

Cloud Hosting - This is often one of the most significant costs. The monthly expenses for hosting your application on a cloud service provider like AWS, Azure, or Google Cloud will need to be estimated. Costs can include virtual machine instances, storage, data transfer, and any additional services you use.

Data Storage - Our application relies on a database, so the monthly expenses for storing and managing our data will be calculated.

Development and Maintenance - A budget for ongoing development, bug fixes, updates, and any additional features or improvements will be allocated. This cost can vary based on our development team's size and work required.

Backup and Disaster Recovery - We will plan for the cost of backup services and disaster recovery solutions to ensure data availability and integrity.

Support and Helpdesk - We offer customer support/a helpdesk, so we'll need to consider the cost of maintaining these services, including staff salaries or any third-party support tools.

# Change Management

The organisation will adopt our software if it sees clear value in it. This value can be in the form of increased efficiency, cost savings, revenue generation, or improved processes. Demonstrating the tangible benefits of your software is key to gaining organizational adoption. If our software aligns with the organisation's strategic goals and objectives, it's more likely to be adopted. Clearly communicate how your software supports the achievement of these goals.

Organisations adopt software to gain a competitive advantage, so they want to stay ahead in the market and provide better services to their customers or users. Our software can provide that edge by possessing features that provide a competitive edge. Our software addresses compliance and security concerns, so organisations are more likely to adopt it.

Our strategy to gain adoption from both the organization and the users is to offer the organisation the opportunity to run pilot programs to test our software's effectiveness; this allows them to see real-world benefits and build trust. Customisation and training will provide customisation options and training services to tailor the software to the organization's specific needs and ensure users can use it effectively.

# Change Management

**Flexible Planning:** Agile project management allows for flexibility in project planning. New requirements or changes in priorities can be integrated into the project without causing major disruptions. The product backlog, which contains a list of all desired features and requirements, can be reprioritized at any time based on business needs.

**Backlog Refinement:** In an Agile environment, the project team regularly reviews and refines the product backlog. This is where new requirements are introduced, existing ones are updated, and unnecessary or outdated items are removed. The product backlog evolves to reflect the most current understanding of project requirements.

**Scope Control:** Agile methodologies emphasize scope control. This means that when new requirements are introduced, the project team works with stakeholders to determine their impact on project scope, timeline, and resources. Trade-offs and adjustments are made, and changes are incorporated in a controlled manner.

**Continuous Improvement:** Agile encourages a culture of continuous improvement. At the end of each sprint, a retrospective is held to review what went well and what can be improved. This iterative feedback loop helps the team identify ways to handle changing requirements more effectively in future sprints.

**Transparency:** Transparency is a core principle of Agile. All stakeholders have visibility into the project's progress and can monitor how requirements are evolving. This transparency fosters trust and collaboration among team members.



# **End of** **Document**

