ARS FOR QANTAS DOMESTIC

SYSTEM ANALYSIS AND DESIGN

JULY 16, 2024

ANESU DAVID PASIPANODYA K200810

LUIZIANE BRANTA K210302

Kent Institute of Australia, Sydney Campus

SAAD - Group Assignment

Table of Contents

Task 1. Understanding the problem

1.1 Task Outline

"Qantas Domestic wants us to assist them in building them an online Airline Reservation System (ARS) that helps the passengers with the entire process from purchasing tickets to checking in. Of course, they still need to drop in their luggage but having a web check in can lessen a lot of hassle for the passengers. Their highest priority is the availability of the system 24/7. Furthermore, they want the components of the system to be changed independently of one another. They plan to incorporate many changes in the coming future, and they are not sure what changes they might make to the system; well, a typical customer who doesn't know what to do but wants to do it anyway. However, we are to design it to make sure their desire is addressed.

For now, the customers will be able to search flight details by providing the source location, desired destination, and date. They will be able to book tickets by providing their travel and contact details. The system will have a registration and login facility for recurring customers, but it is not necessary for booking tickets or any of the activities. The reservation would be kept for 24 hours before confirming it with payment. The payment option should be available when they book the ticket. Any payment received for the booked tickets will be counted as confirmed. The confirmed tickets can be cancelled provided that the customer request cancellation at least 24 hours before the flight. The cancellation of the tickets will be verified by the staff. The ticket confirmed will be sent to the customer's email. The staff will be able to confirm ticket by overriding the payment option. A web check in part will be activated 12 hours before the flight by the staff and the customers who have purchased the tickets can check in and print their boarding pass. Since issuing boarding pass is a security issue, we are required to tackle it by connecting with the immigration portal and verifying the travel documents provided by the customer. If failed, the boarding pass will not be issued, and the designated customer will be contacted by the staff manually. The registered users will be allocated discounts by the staff members for their airtime which they will have an option to avail during booking tickets."

Based on this we can extract that the Airline Company (QANTAS) requires us to make an ARS System with the following qualities: -

1.1.1. **24/7 Availability:**

We need to make sure the system works all the time without interruptions

1.1.2. **Modular Design**:

We are required to design the system so that parts can be changed easily in the future.

1.1.3. **Search Flights:**

Let customers search for flights using the source, destination, and date.

1.1.4. **Book Tickets:**

Allow customers to book tickets by entering travel and contact details.

Include optional registration and login for frequent customers.

1.1.5. **Reservation and Payment:**

The system needs to:

- Hold reservations for 24 hours.
- Provide a payment option during booking.
- Confirm reservations once payment is received.
- Allow staff to confirm tickets without payment if needed.

1.1.6. **Ticket Confirmation and Cancellation:**

Send confirmed tickets to customers via email.

Allow customers to cancel tickets at least 24 hours before the flight.

Have staff verify cancellations.

1.1.7. **Web Check-In:**

Activate web check-in 12 hours before the flight.

Enable customers to check in online and print boarding passes.

1.1.8. **Security and Verification:**

Verify travel documents with the immigration portal before issuing boarding passes.

Manually contact customers if document verification fails.

1.1.9. **Discounts:**

Allow staff to give discounts to registered users.

Let customers use discounts when booking tickets.

Possible routes and solutions for Implementation: -

1.1.1 ***24/7 Availability:***

- *We need to make sure the system works all the time without interruptions*

Proposal:

To ensure 24/7 availability for the Qantas Domestic ARS, we will have to find a way to implement robust cloud-based infrastructure, incorporating factors like redundancy, (Aktas et al. 2021) load balancing, and automated failover mechanisms (PR Newswire 2022). Our team will have to find a way to do continuous monitoring and proactive maintenance will ensure minimal downtime, while scalable architecture will handle varying loads and ensure seamless user experience around the clock. (Saini et al., 2023)

1.1.2. ***Modular Design:***

- *We are required to design the system so that parts can be changed easily in the future.*

Proposal:

To achieve modularity, we would have to develop a system design which can be later adjusted or has room for future adjustments for the Qantas Domestic ARS. We will have to consider using a microservices architecture (Krämer, Frese & Kuijper 2019). Each component could be independently deployable and maintainable, facilitating easy updates and future changes without affecting the entire system. This ensures flexibility and scalability for ongoing enhancements. separating functions like flight search, booking, payment, and user management into distinct services. Each service will have its own API, enabling independent updates and scaling. (Dogra et al. 2022)

1.1.3. ***Search Flights:***

- *Let customers search for flights using the source, destination, and date.*

Proposal:

To enable customers to search for flights, we will have to develop a more robust search engine and integrate it within the ARS system's database. The search engine will have to filter flights based on location of the user, their destination, and date/times of arrival and departure. Consistent collaboration with database administrators and front-end developers ensures accurate and user-friendly search results. Ongoing testing will ensure search experience is optimised and easy to use for students. (PR Newswire 2023)

Collaboration:

- Database Administrators: Ensure data accuracy and query efficiency.
- Front-End Developers: Design a user-friendly interface.
- QA Testers: Conduct usability testing and refinement.

1.1.4. ***Book Tickets:***

- *Allow customers to book tickets by entering travel and contact details. Include an optional registration and login for frequent customers.*

Proposal:

We could implement a booking system that allows the customer to enter their unique travel and contact details. Optional registration and login for recurring customers could be added in the system. Working with back-end developers for data handling and front-end developers for the user interface will have to be crucial to implementation. We'd have to ensure secure data transmission and storage as this is paramount to our success. (Bukhari & Kim 2012)

Collaboration:

- Back-End Developers: Handle data processing and storage.
- Front-End Developers: Design intuitive booking forms.
- Security Experts: Ensure data security and compliance.

1.1.5. ***Reservation and Payment***:

- *The system needs to:*
  - *Hold reservations for 24 hours.*
  - *Provide a payment option during booking.*
  - *Confirm reservations once payment is received.*
  - *Allow staff to confirm tickets without payment if needed.*

Proposal:

The system will have to hold reservations for 24 hours, confirm them upon payment, and allow staff overrides at the backend. Integrating secure payment gateways and developing a reservation management system will be essential to completing this as we cannot incur any data breaches or security-based repercussions. Collaborating with payment gateway providers, back-end developers, and staff trainers ensures smooth operations.

Collaboration:

- Payment Gateway Providers**:** Integrate secure payment options.
- Back-End Developers: Develop reservation and payment handling.
- Staff Trainers: Train staff on system overrides

1.1.6. ***Ticket Confirmation and Cancellation***:

- *Send confirmed tickets to customers via email.*
- *Allow customers to cancel tickets at least 24 hours before the flight.*
- *Have staff verify cancellations.*

Proposal:

Confirmed tickets will have to be emailed to customers, who can cancel tickets at least 24 hours before the flight. QANTAS Staff will have the ability to verify cancellations. Developing automated email systems and a user-friendly cancellation process, with collaboration between front-end developers, email service providers, and customer support teams, to ensure efficiency.

Collaboration:

- Front-End Developers: Design the cancellation interface.
- Email Service Providers: Integrate automated email confirmations.
- Customer Support Teams: Manage and verify cancellations

1.1.7. ***Web Check-In***:

- *Activate web check-in 12 hours before the flight.*
- *Enable customers to check in online and print boarding passes.*

Proposal:

Web check-in will have to be activated 12 hours before the flight, allowing customers to check in online and print boarding passes. Collaborating with front-end developers for the interface, back-end developers for data processing, and staff for activation ensures seamless check-in experiences.

Collaboration:

- Front-End Developers: Develop the check-in interface.
- Back-End Developers: Handle check-in data processing.

1.1.8. ***Security and Verification***:

- *Verify travel documents with the immigration portal before issuing boarding passes.*
- *Manually contact customers if document verification fails.*

Proposal:

To secure boarding passes, we will need to connect with the immigration portal for document verification. Manual contact for failed verifications ensures the security of data. Collaborating with immigration authorities, back-end developers, and security experts would ensure the most robust verification processes.

Collaboration:

- Immigration Authorities: Integrate document verification.
- Back-End Developers: Develop verification processes.
- Security Experts: Ensure the integrity and security of verifications.

1.1.9 ***Discounts****:*

- *Allow staff to give discounts to registered users.*
- *Let customers use discounts when booking tickets.*

Proposal:

Staff will allocate discounts to registered users based on their airtime, with options to use discounts during booking. Developing discount management systems and ensuring seamless integration with booking processes will be key. Collaborating with back-end developers, front-end developers, and marketing teams ensures effective discount usage.

Collaboration:

- Back-End Developers: Develop discount management systems.
- Front-End Developers: Integrate discount options in booking.
- Marketing Teams: Define discount strategies and policies. *

**These are only rough paths that could potentially be used for implementations to give an idea of the scope of the project, but it can all be discussed and dissected at length during project introduction*

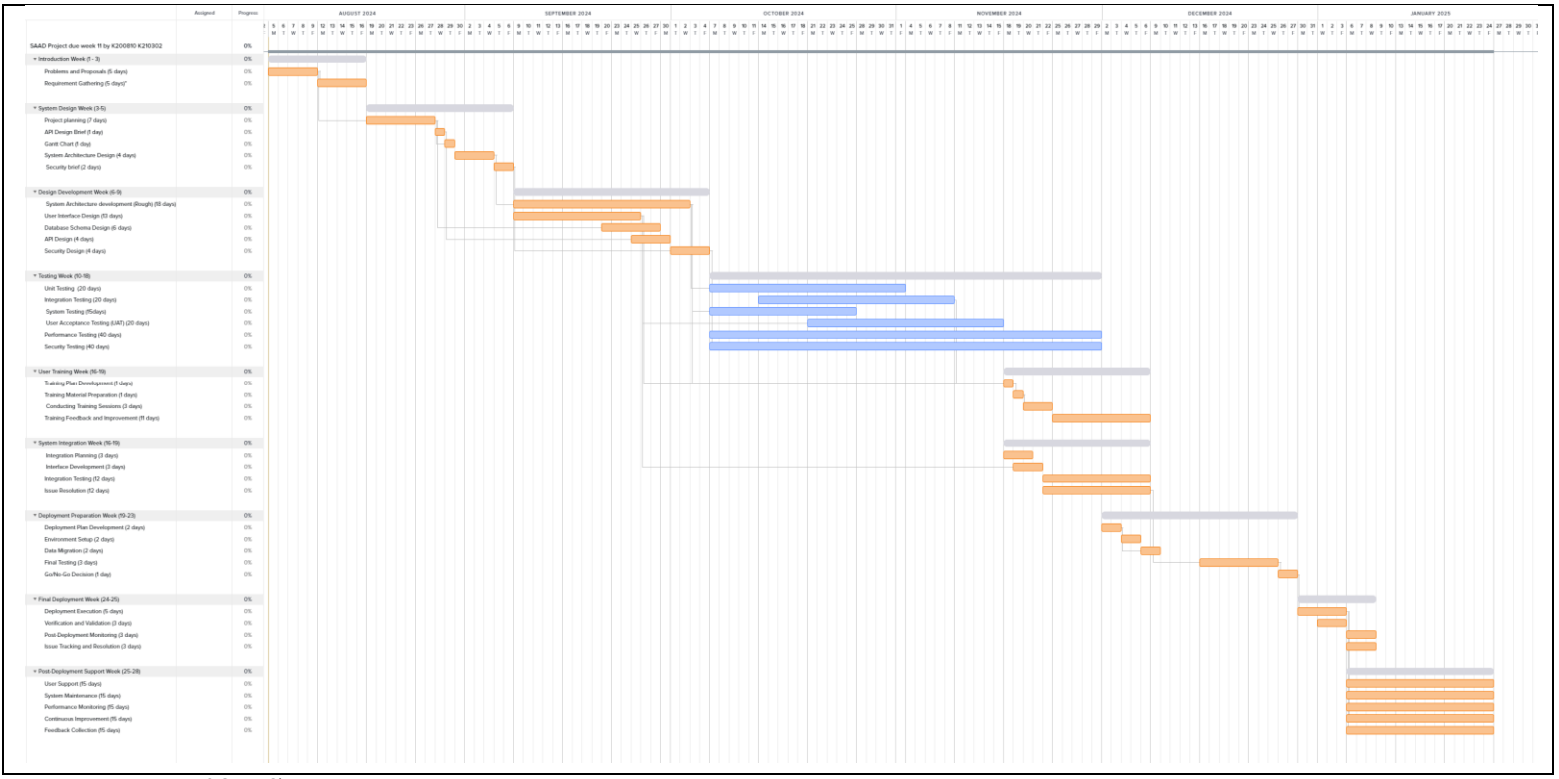
FIG 1.1

1.2 Dependencies, Critical Path and Non-Critical Pathways to completing the task.

To find out the Dependencies, Critical Path and Non-Critical Pathways to completing the task, we are going to use a Gannt Chart to track the Critical Path and non-Critical Path. And then we will use this info to calculate slack time.

Gannt Chart

To complete the Gannt chart for this we are going to use TeamGannt.



1.2 Gannt Chart

This is a project Timeframe table with the dependencies outlined

Task	Description	Start Week	Duration (in weeks)	End week	Dependencies
I	**Introduction** i) Problems and Proposals (5 days) * ii) Requirement Gathering (5 days)	1	2 (10 days)	2	**I** ii) is dependent on **I** i)
II	**System Design** i) Project planning (7 days) ii) API Design brief (1 days) iii) Gannt Chart (1 day) iv) Security brief (4 days) v) System Architecture Design (2 days)	3	3 (15 days)	5	**II** i) is dependent on **I** ii). **II** ii) is dependent on **II** i) and **II** iii is dependent on **II** ii
III	**Design Development** i) System Architecture (20 days) ii) User Interface Design (13 days) iii) Database Schema Design (6 days) iv) API Design (4 days) v) Security Design (4 days)	6	4 (20 days)	9	**III** i) is dependent on **II** iv). **III** iii) and **III** iii) are dependent on **II** v). **III** iv) is dependent on **I**iii)
IV	**Testing** i) Unit Testing (20 days) ii) Integration Testing (20 days) iii) System Testing (15 days) iv) User Acceptance Testing (UAT) (30 days) v) Performance Testing (40 days) vi) Security Testing (40 days)	10	8 (40 days)	18	**IV** vi) is dependent on **III** v) **IV** i) ii) and iii) are dependent on **III** i)
V	**User Training** i) Training Plan Development (1 days) ii) Training Material Preparation (1 days) iii) Conducting Training Sessions (3 days) iv) Training Feedback and Improvement (11 days)	16	3 (15 days)	19	**V** ii) is dependent on **V** i) **V** i) is dependent on **IV** iv) and **III** i) and ii)
VI	**System Integration** i) Integration Planning (3 days) ii) Interface Development (3 days) iii) Integration Testing (12 days) iv) Issue Resolution (12 days)	16	3 (20 days)	19	
VII	**Deployment Preparation** i) Deployment Plan Development (2 days) ii) Environment Setup (2 days) iii) Data Migration (3 days) iv) Final Testing (3 days) v) Go/No-Go Decision (1 days)	19	2 (10 days)	23	**VII** iv) is dependent on **VI** iv) **VII** iii is dependent on **VII** ii **VII** v is dependent on **VII** iv
VIII	**Final Deployment** i) Deployment Execution (5 days) ii) Verification and Validation (3 days) iii) Post-Deployment Monitoring (3 days) iv) Issue Tracking and Resolution (3 days)	24	2 (8 days)	25	**VIII** i) is dependent on **VII** v) **VIII** iv) is dependent on **VII** i)

IX	**Post-Deployment Support** i) User Support (15 days) ii) System Maintenance (15 days) iii) Performance Monitoring (15 days) iv) Continuous Improvement (15 days) v) Feedback Collection (15 days)	25	3 (15 days)	28	All of **IX** is dependent on **VIII** i). They all run simultaneously

Fig1.3

*All days are weekdays. Weekends are excluded

Project will Start from the beginning of September 2024 and will run for 28 weeks up until the beginning of 2025.

Dependencies

I ii) is dependent on I i)

II i) is dependent on I ii). II ii) is dependent on II i) and II iii) is dependent on II ii)

III i) is dependent on II iv). III iii) and III iii) are dependent on II v). III iv) is dependent on Iiii)

IV vi) is dependent on III v)

IV i) ii) and iii) are dependent on III i)

V ii) is dependent on V i)

V i) is dependent on IV iv) and III i) and ii)

VII iv) is dependent on VI iv)

VII iii) is dependent on VII ii)

VII v) is dependent on VII iv)

VIII i) is dependent on VII v)

VIII iv) is dependent on VII i)

All of IX is dependent on VIII i)

Critical Path Analysis

As a result, the Critical Path of the project is as follows:

I Introduction

ii) Problems and Proposals (5 days)

II System Design

i) Project Planning (7 days)

ii) API Design Brief (1 day)

iv) Security Brief (4 days)

III Design Development

i) System Architecture (20 days)

ii) User Interface Design (13 days)

iii) Database Schema Design (6 days)

iv) API Design (4 days)

v) Security Design (4 days)

IV Testing

ii) Integration Testing (20 days)

iii) System Testing (15 days)

iv) User Acceptance Testing (UAT) (30 days)

v) Performance Testing (40 days)

vi) Security Testing (40 days)

V User Training

i) Training Plan Development (1 day)

ii) Training Material Preparation (1 day)

VI System Integration

i) Integration Planning (3 days)

iv) Issue Resolution (12 days)

VII Deployment Preparation

i) Deployment Plan Development (2 days)

iv) Final Testing (3 days)

v) Go/No-Go Decision (1 day)

VIII Final Deployment

i) Deployment Execution (5 days)

This means that using the Gannt Chart the total no. of days for the critical path to take place would be:

7+1+40+7+5= 60 ***days*** or ***12 weeks*** in our calendar

The non-Critical Path would be 5x28=140days or 28 weeks

Slack Time

Slack time, also known as float, represents the amount of time a task can be delayed without affecting the project's overall timeline. For our ARS in question here is a list of the detailed description of the slack time for various tasks:

Given the formula of slack time being Time for critical path – Time for non-critical path Slack time = 140-60=80 days, the number being increased significantly if we account for the post deployment support of 15 days. Without post deployment support it becomes 65 days of total slack time

Summary

Critical Path: 60 days or 12 weeks

Non-Critical Paths: Slack is 65 days or 13 weeks

Task 2. Modelling Tasks

2.1. Case description Qantas Domestic System.

In this section, we describe the Qantas Domestic ARS using case diagrams created with

Draw.io. The use case diagram illustrates the various interactions between the system and its users, detailing the functional requirements of the system. This includes the processes of logging in, searching for flights, booking tickets, making payments, and other critical user activities. By illustrating these interactions, we gain a clear understanding of the system's scope, and the user roles involved in its operation.

Use Case Diagram

Use Case Description Examples.

To Login

Actors: Customer

Preconditions: Customer has created an account with Qantas by providing email, number and making a password

Postconditions: Customer is taken to initial search page.

Description: Customer enters their details

To Logout

Actors: Customer

Preconditions: Customer has created an account with Qantas by providing email, number and making a password, but now wants to logout

Postconditions: Customer is logged out of the account.

Description: Customer goes to logout of their account

To Search Flights

Actors: Customer

Preconditions: Customer provides source destination end destination, and to and from dates.

Postconditions: Flight details are displayed.

Description: Customer searches for flights using specified criteria

To Book Tickets

Actors: Customer
Preconditions: Customer selects a flight and provides travel and contact details.
Postconditions: Ticket is booked, and the reservation is created.
Description: Customer books a ticket for a selected flight.

To make Payments

Actors: Customer
Preconditions: Customer has a booked reservation.
Postconditions: Payment is processed, and reservation is confirmed.
Description: Customer makes a payment for the booked ticket.

To Cancel tickets

Actors: Customer, Airline Staff
Preconditions: Ticket is confirmed, and cancellation requests are made 24 hours before the flight.
Postconditions: Ticket is cancelled after staff verification.
Description: Customer requests ticket cancellation, verified by staff.

To Submit Immigration Documents

Actors: Customer,
Preconditions: Customer has a booked ticket and valid travel documents according to the requirements from the Immigration (to ensure they're allowed to travel).
Postconditions: Documents are submitted for verification by Immigration.
Description: Customer submits travel documents for immigration verification.

To receive boarding pass

Actors: Customer
Preconditions: Ticket is confirmed, and documents are verified by immigration.
Postconditions: Boarding pass is issued.
Description: Customer receives a boarding pass post verification.

Web Check-in

Actors: Customer, Airline Staff
Preconditions: Ticket is confirmed, and check-in is activated 12 hours before the flight.
Postconditions: Boarding pass is printed.

Description: Customer checks in online and prints boarding pass.

Verify Cancellation

Actors: Airline Staff
Preconditions: Customer requests ticket cancellation.
Postconditions: Cancellation is confirmed.
Description: Staff verifies and confirms ticket cancellation.

Override Payment

Actors: Airline Staff
Preconditions: Special circumstances requiring staff intervention.
Postconditions: Ticket is confirmed without payment.
Description: Staff confirms ticket by overriding payment.

Manage Discounts

Actors: Airline Staff
Preconditions: Special circumstances requiring staff intervention.
Postconditions: Ticket is confirmed without payment.
Description: Staff confirms ticket by overriding payment.

Verify Documents

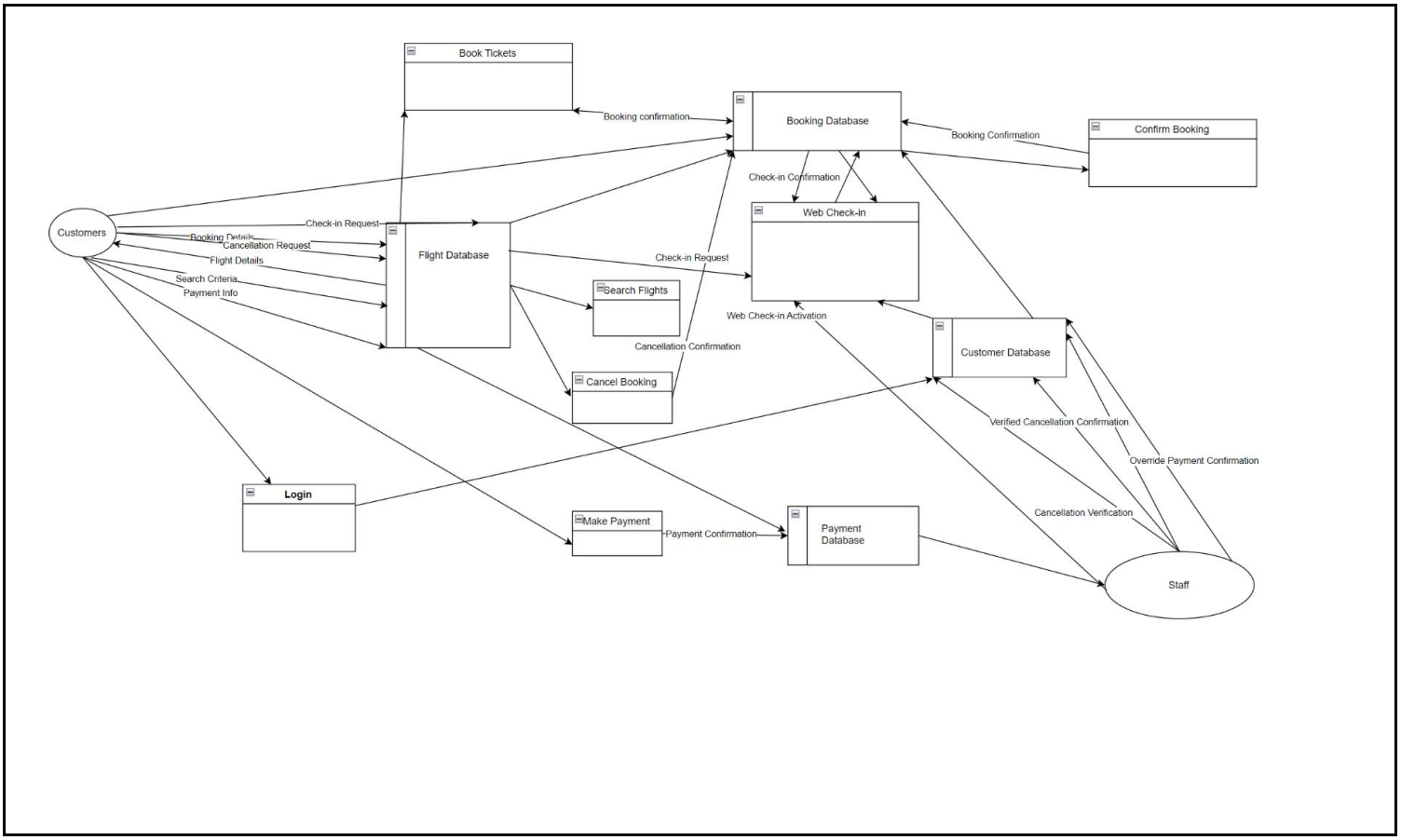
Actors: Immigration, Airline Staff
Preconditions: Customer submits travel documents.
Postconditions: Documents are verified, and the customer is informed.
Description: Immigration and staff verify customer travel documents.

Cancel Tickets

Actors: Customer, Airline Staff
Preconditions: Ticket is booked, and customer wants to cancel it.
Postconditions: Ticket is cancelled if conditions are met.
Description: Customer cancels the ticket within allowed timeframe.

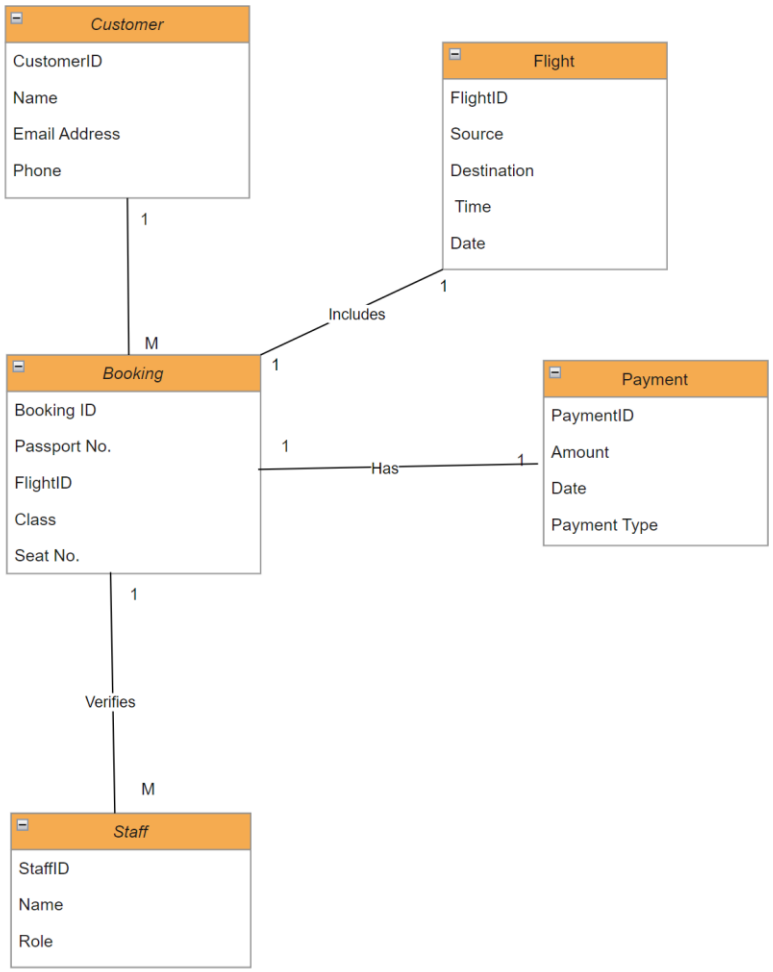
2.2 Qantas Domestic ARS DFD 1 System.

In this diagram we have represented the Data Flow between processes, data stores, and external entities for the Qantas Domestic ARS using Draw.io. The DFD visually represents the flow of information within the system, illustrating how data is processed and transferred between different components. This diagram will help us to understand the system's functionality and the interaction between various processes and data stores.



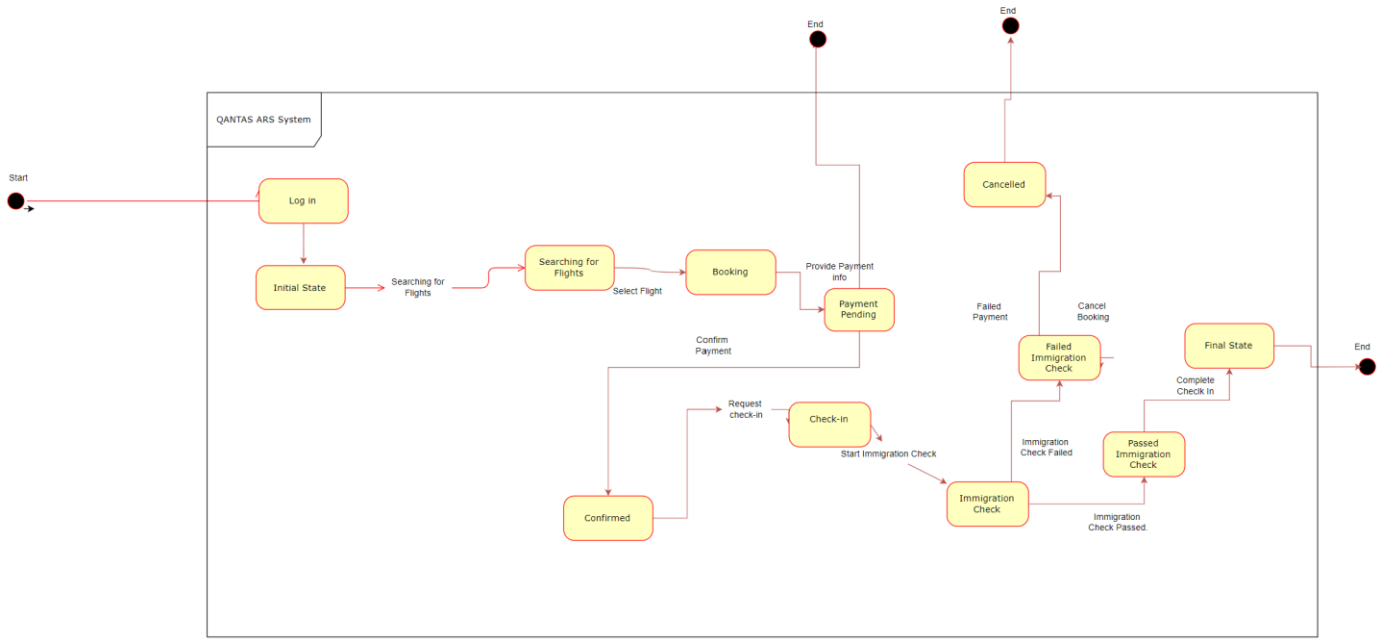
2.3 Qantas Domestic ARS ERD diagram.

This diagram represents the Relationship between entities for the Qantas Domestic ARS. The ERD outlines the system's data structure by showing the entities involved, their attributes, and the relationships between them. This diagram is essential for understand the system's functionality and the interaction between various processes and data stores.



Customer makes a booking. Booking includes flight details and has payment information and is verified by staff.

2.4 Qantas Domestic System State Machine Diagram.



Task 3. System Development Method of Choice: *Agile Methodology*.

To have the capacity to adapt to the fast-changing requirements of the local aviation industry where Qantas hopes to compete, we chose to adopt the ***Agile*** (Scrum) methodology for Qantas Domestic's Airline Reservation System (ARS) development project. We've made this decision because it gives us room to be more flexible, to adapt to aspects of the project at every level and at any time, and to mitigate the risk of major project developmental changes. These tend to happen further along the project, and with the project being done in sprints, we can mitigate this by consulting stakeholders at every level during the scrums where we meet with them. This decision has been made because of the project requirements, as the Agile method has a more iterative approach than the other options, such as SDLC Waterfall. Certain parts of the project will require incremental delivery, such as the design phase, with changes most probably being made at these phases during the development. Independent and empowered teams are also likely to cover work faster as their independence will most likely promote a greater work ethic and more technical decision-making, in, for example, the API or database schema design, which requires specialist programmers and developers. (Venkatesh et al., 2020) As a result of the complexity of this project and its necessity to be user-friendly, the ability to create a Minimal Viable Product (MVP) may be crucial during the UAT phase (IV iv), as this would enhance this aspect of the project. Lean thinking also needs to be encouraged because of this project, so this benefit of the Agile method is another advantage of using this Agile methodology to complete our project. We cannot use the SDLC, as an example of another type of method, for the following reasons:

Flexibility and Adaptability: SDLC has a more rigid and sequential structure.

Customer Involvement: Agile involves continuous customer feedback, unlike SDLC, which has limited customer interaction after initial requirements.

Speed of Delivery: Agile delivers functional parts incrementally, while SDLC delivers the entire product at the end.

Risk Management: SDLC has higher risks due to late issue discovery.

Team Collaboration and Communication: SDLC often involves siloed teams.

Response to Market Changes: SDLC is slower to respond to market changes.

Project Visibility and Transparency: SDLC has limited visibility until later stages.

Continuous Improvement: Agile focuses on continuous improvement through regular retrospectives, unlike SDLC, which places less emphasis on this.

Therefore, we are going to use the Agile (Scrum) developmental methodology, and using this, we will be in a more suitable position to complete the project.

References

- Aktas, MF, Far, AB, Soljanin, E & Whiting, P 2021, 'Evaluating load balancing performance in distributed storage with redundancy', *IEEE Transactions on Information Theory*, vol. 67, no. 6, pp. 3623–3644, viewed 25 July 2024, https://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=150448700&site=ehost-live.

- Bukhari, AC & Kim, Y-G 2012, 'Integration of a secure type-2 fuzzy ontology with a multi-agent platform: A proposal to automate the personalized flight ticket booking domain', *Information Sciences*, vol. 198, pp. 24–47, viewed 26 July 2024, https://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=74408914&site=ehost-live.

- Cooper, RG & Sommer, AF 2020, 'New-product portfolio management with Agile: Challenges and solutions for manufacturers using Agile development methods', *Research Technology Management*, vol. 63, no. 1, pp. 29–38, viewed 26 July 2024, https://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=144827045&site=ehost-live.

- Dogra, A, Mahna, S, Padhee, SS & Singla, E 2022, 'Unified modelling of unconventional modular and reconfigurable manipulation system', *Robotics & Computer-Integrated Manufacturing*, vol. 78, p. N.PAG, viewed 26 July 2024, https://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=157991782&site=ehost-live.

- Draw.io 2024, *Diagram software and flowchart maker*, available at: https://www.draw.io/ [Accessed 1 August 2024].

- Eby, K 2016, 'Understanding Agile software development lifecycle and process workflow', *Smartsheet*, available at: https://www.smartsheet.com/understanding-agile-software-development-lifecycle-and-process-workflow.
- Kerzner, H 2017, *Project management: A systems approach to planning, scheduling, and controlling*, John Wiley & Sons.

- King, T 2011, 'Slack scheduling', *EDN*, vol. 56, no. 16, pp. 29–33, viewed 26 July 2024, https://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=65240239&site=ehost-live.

- Krämer, M, Frese, S & Kuijper, A 2019, 'Implementing secure applications in smart city clouds using microservices', *Future Generation Computer Systems*, vol. 99, pp. 308–320, viewed 26 July 2024, https://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=139747254&site=ehost-live.

- Kramer, JD & Wagner, TJ 2019, 'Developmental test and requirements: Best practices of successful information systems using Agile methods', *Defense Acquisition Research Journal: A Publication of the Defense Acquisition University*, vol. 26, no. 2, pp. 128–151, viewed 26 July 2024, https://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=136595302&site=ehost-live.

- Mamakou, XJ 2023, 'Intentions to continue using agile methods: The case of the Greek banking sector', *Journal of Systems & Software*, vol. 202, p. N.PAG, viewed

26 July 2024, https://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=163948179&site=ehost-live.

- Meredith, JR & Mantel, SJ 2011, *Project management: A managerial approach*, John Wiley & Sons.

- PR Newswire 2022, 'Buoyant announces automated multi-cluster failover capabilities in Linkerd', *PR Newswire US*, 9 March, viewed 26 July 2024, https://search.ebscohost.com/login.aspx?direct=true&db=bwh&AN=202203090800PR.NEWS.USPR.NY86305&site=ehost-live.
- PR Newswire 2023, 'Login Enterprise introduces new features to accelerate root cause analysis and improve end-user experience for VDI and DaaS', *PR Newswire US*, 22 May, viewed 26 July 2024, https://search.ebscohost.com/login.aspx?direct=true&db=bwh&AN=202305220800PR.NEWS.USPR.NE06022&site=ehost-live.

- Saini, M, Maan, VS, Kumar, A & Saini, DK 2023, 'Cloud infrastructure availability optimization using Dragonfly and Grey Wolf optimization algorithms for health systems', *Journal of Intelligent & Fuzzy Systems*, vol. 45, no. 4, pp. 6209–6227, viewed 26 July 2024, https://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=173420166&site=ehost-live.

- TeamGantt 2024, *TeamGantt*, available at: https://www.teamgantt.com [Accessed 2 August 2024].

- Venkatesh, V, Thong, JYL, Chan, FKY, Hoehle, H & Spohrer, K 2020, 'How agile software development methods reduce work exhaustion: Insights on role perceptions and organizational skills', *Information Systems Journal*, vol. 30, no. 4, pp. 733–761, viewed 26 July 2024, https://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=143678239&site=ehost-live.