UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Department of Computer Science
Faculty of Engineering, Built Environment & IT
University of Pretoria

COS301 - Software Engineering

Office Booker

Team name: Kryptos Kode

Name	Student Number	Email
Ying Hao Li*	u20460687	u20460687@tuks.co.za
Arul Agrawal	u18053239	u18053239@tuks.co.za
Grant Bursnall	u15223893	u15223893@tuks.co.za
Damian Vermeulen	u20538945	u20538945@tuks.co.za
Brett du Plessis	u19037717	u19037717@tuks.co.za

* - Team Leader

Contents

1 Introduction **4**
1.1 Vision . . . 4
1.2 Objectives . . . 4

2 Class Diagram **5**

3 User Characteristics **6**
3.1 Employee . . . 6
3.1.1 Purpose . . . 6
3.1.2 Technical Skills . . . 6
3.1.3 Experience . . . 6
3.1.4 User Story . . . 6
3.2 Administrators . . . 6
3.2.1 Purpose . . . 6
3.2.2 Technical Skills . . . 6
3.2.3 Experience . . . 6
3.2.4 User Story . . . 6
3.3 Guests . . . 7
3.3.1 Purpose . . . 7
3.3.2 Technical Skills . . . 7
3.3.3 Experience . . . 7
3.3.4 User Story . . . 7

4 Requirements **7**
4.1 Use Cases . . . 7
4.2 Functional Requirements . . . 9
4.3 Non-Functional Requirements . . . 9
4.4 Subsystems . . . 10
4.4.1 Deleting a Booking: U1 . . . 10
4.4.2 Adding a Booking: U2 . . . 10
4.4.3 Viewing an Office Space: U3 . . . 10
4.4.4 Mapping out Office Space: U4 . . . 11
4.4.5 Real Time updates of the Office space slots availability: U5 . . . 11
4.4.6 Prevent Users that are not employees from making Bookings: U6 11

5 Quality Requirements **12**
5.1 Availability Q1 . . . 12
5.2 Responsiveness Q2 . . . 12
5.3 Reliability Q3 . . . 12
5.4 Maintainability Q4 . . . 12
5.5 Usability Q5 . . . 12

6 Trace-ability Matrix **13**

7 Architecture **14**

8 Deployment Model **15**

1 Introduction

1.1 Vision

Office Booker is a system that allows a user to interactively map out an office space and then book conference rooms and desks within the mapped-out office space. Many employees began working from home when the COVID-19 pandemic hit and thus a system is needed to help with organising who uses the limited office space and this is done through booking the desired space for the desired time.

1.2 Objectives

We, as Kryptos Kode, need to work diligently for the desired goal of creating the office Booker with a certain quality standard in place. By each sprint we should aim to have more fully functioning core requirements ready as well as some non-core requirements ready for the project to be completed on time with the necessary features implemented.

- The Office Booker should have a visual representation of the office for mapping out the space and its available resources.
- A way of selecting and booking slots in the office for use.
- A method of preventing other users that are not employees of the company from using the Booker for the company's office space.
- Able to view when spaces are available over a certain period of time.
- An office wide schedule view of bookings.
- Real time updates on the office slots availability.

Meeting these requirements will be the goal for creating the clients desired Office Booker

2 Class Diagram

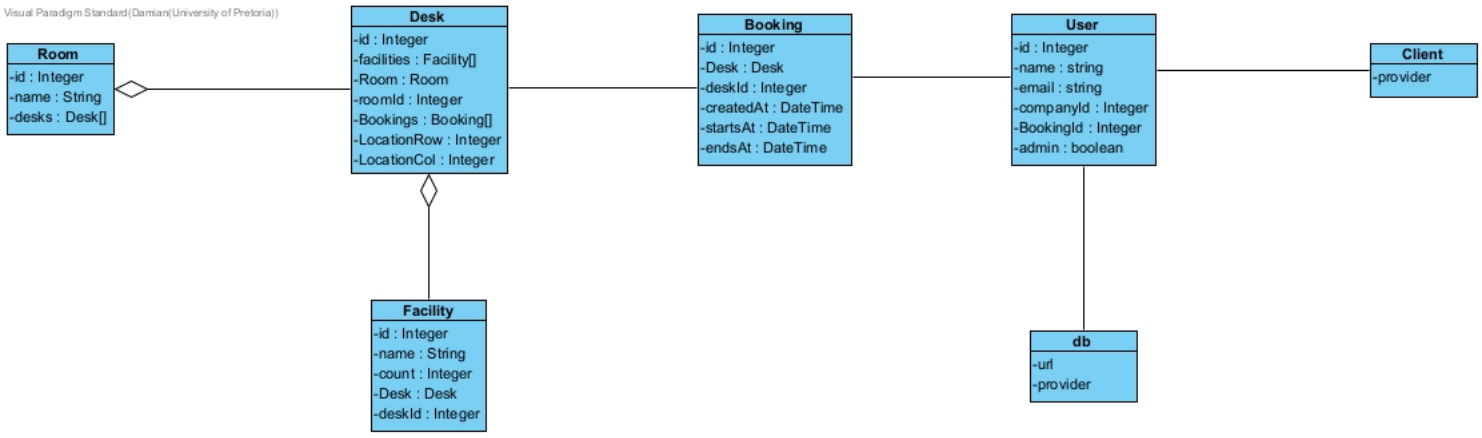

Figure 1: Class Diagram

3 User Characteristics

3.1 Employee

3.1.1 Purpose

An employee can book a slot to use in the office, book a meeting room, manage bookings and check availability of bookings.

3.1.2 Technical Skills

- Internet literate to search the web application and use it for bookings.

3.1.3 Experience

Minimal experience required to use this app.

3.1.4 User Story

An employee would want to go in to the office but could find difficulty as others are trying to go in at the same time and there may not be enough room. Using the Office Booker the employee can securely select when what place they want to have in the office without causing any clashes. An employee could also decide that they want to change their bookings and go to remove their booking and create a new one.

3.2 Administrators

3.2.1 Purpose

The admins can use this to delete others bookings and make their own bookings like an employee could as well as allow bookings for guests.

3.2.2 Technical Skills

- The basics of internet related skills to search the web application and use it for bookings.
- Understanding of data and user management

3.2.3 Experience

Experience in administration to properly manage users bookings and allow guest bookings.

3.2.4 User Story

An Administrator of the Office Booker could see someone who does not honour their bookings is continuously making bookings and could choose to remove the employee who does not honour their bookings slot from them to prevent them from wasting a slot. An Administrator would also want to create their own bookings for the office and select a slot for a date and time for them specifically.

3.3 Guests

3.3.1 Purpose

Guests can request a slot to book that must be approved by an admin.

3.3.2 Technical Skills

- Internet literate to search the web application and use it for bookings.

3.3.3 Experience

Minimal experience required to use this app.

3.3.4 User Story

A Guest could want to come to the office but is not an employee in the company. The guest could make a request to the company to allow them to use a slot or room for when the guest makes a visit to the office. An Administrator could grant the request and the guest may now use their booked slot in the office. A guest however would need to cancel their booking through an administrator as a guest does not need an account to request a booking.

4 Requirements

4.1 Use Cases

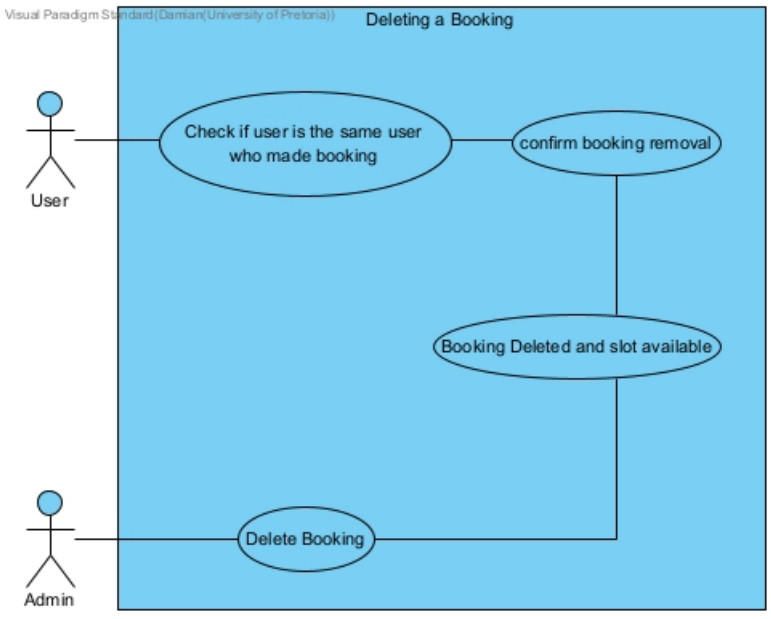

Figure 2: Deleting a Booking

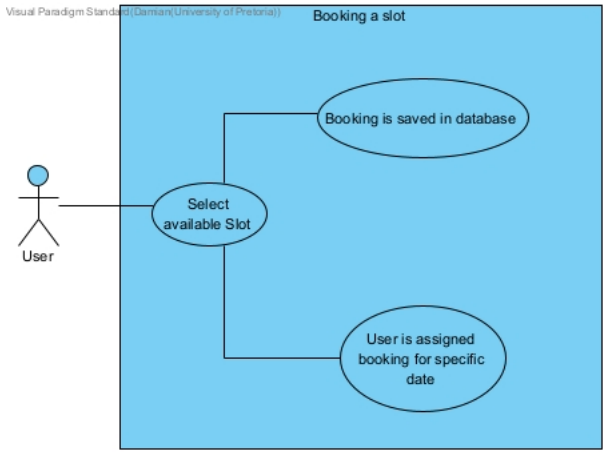

Figure 3: Adding a Booking

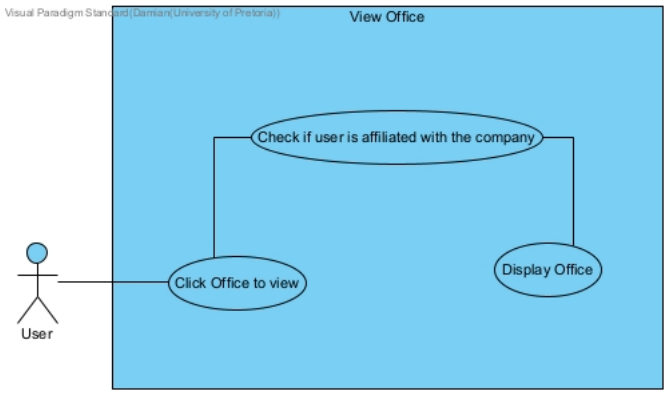

Figure 4: Viewing an Office Space

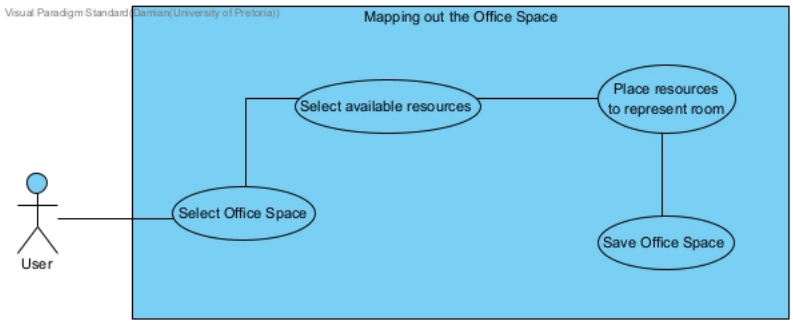

Figure 5: Mapping out Office Space

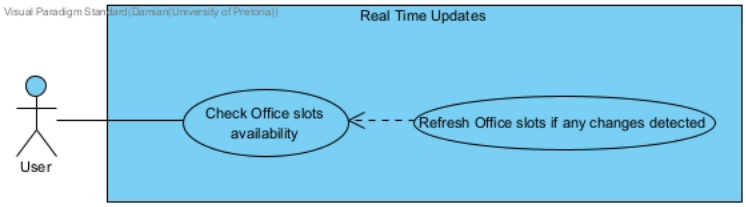

Figure 6: Real Time updates of the Office space slots Availability

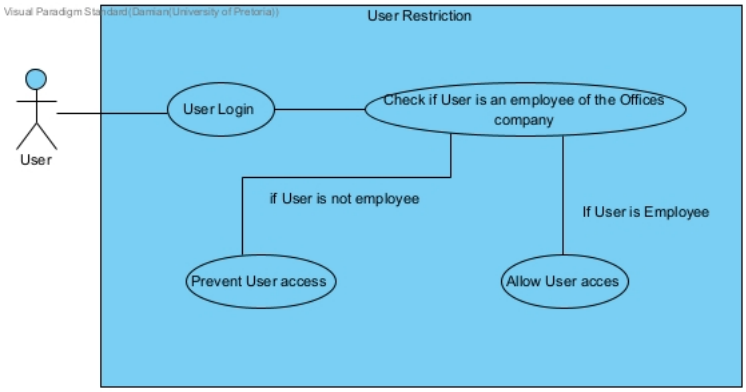

Figure 7: Prevent Users that are not employees from making Bookings

4.2 Functional Requirements

- FR1: A visual representation of the office to map out its resources IE desks, monitors, meeting rooms.
- FR2: A way of selecting and booking slots in the office for use.
- FR3: A method of preventing other users that are not employees of the company from using the Booker for the company's office space.
- FR4: Able to view when spaces are available over a certain period of time.
- FR5: An office wide schedule view of bookings.
- FR6: Real time updates on the office slots availability.

4.3 Non-Functional Requirements

- NFR1: The ability to allow users to create a guest account to book.
- NFR2: The ability to create an office layout using svgs with a drag and drop system.
- NFR3: The ability to rate employees on their rate of honouring their bookings.
- NFR4: The ability to book a slot and send invites to work colleagues to join the slot.

4.4 Subsystems

4.4.1 Deleting a Booking: U1

The system can delete a booking when an admin chooses to remove it or the employee who made the booking can delete it. It uses the same data as the user adding a booking. The user should be able to select a booking and remove their own booking unless they are an admin in which they can remove any users booking. It uses the following data:

- Booking Id
- Date
- Desk Id
- Employee Id

4.4.2 Adding a Booking: U2

The system can add a booking when the slots have been selected for a specific date and time that a user selects. It uses the following data:

- Booking Id
- Date
- Desk Id
- Employee Id

The user selects a slot and it is added to the system for a specific date for the user to use in the office.

4.4.3 Viewing an Office Space: U3

The system allows users to view the office space and its resources. The system shows all bookings made and all slots that are still available for users to make use of. The System will create an overview of the office space so users can get a clear view of what each section of the office space as well as the availability of its slots and offices. It uses the following data:

- Office Space
- Desks
- Room
- Company

4.4.4 Mapping out Office Space: U4

The system allows for the ability to map out an office space and its resources to allow users to make bookings. The maps can be created in advance by the team for the users to test out and use and users can create their own maps as well. It uses the following data:

- Office Space

4.4.5 Real Time updates of the Office space slots availability: U5

When a booking occurs whilst another user is viewing a slot, the availability should update in real time to show that someone has booked the slot. It uses the following data:

- Office Space
- Desk Id
- Date

4.4.6 Prevent Users that are not employees from making Bookings: U6

If a user is not an employee of a specific company, they cannot make bookings for that company's office space. An administrator must allow grant requests for office spaces to be used by non-employees. It uses the following data:

- Office Space
- Employee Id

5 Quality Requirements

5.1 Availability Q1

The Office Booking system must have an availability of 99% so that user may at almost anytime have the system ready in case of creating, deleting or checking a booking. Users should be given the opportunity to use the system whenever they wish to use it.

5.2 Responsiveness Q2

The Office Booking system should be responsive when making a booking to allow users to create or delete a booking whenever they please with no issues. The system must respond when a user is attempting to make a booking to prevent other users from making a booking over the current users current booking.

5.3 Reliability Q3

The system must be reliable for users. Whenever a user uses the Office Booker they should not be worried about bugs or other issues that could reduce the reliability of the Office Booker.

5.4 Maintainability Q4

Maintainability is a standard that must be kept for the office booking system. If a component or system fails it must then be fixed or reset to a set condition.

5.5 Usability Q5

The Office Booker must apply usability. The user being able to use the system with all of its features will allows for a better user experience, therefore the system must be implemented with care that the user can make use of the system without restriction to how the core features can be used.

6 Trace-ability Matrix

-	U1	U2	U3	U4	U5	U6
Q1	X	X	X	X	X	X
Q2		X	X		X	X
Q3	X	X	X	X	X	X
Q4	X	X	X	X	X	X
Q5	X	X	X	X	X	X

7 Architecture

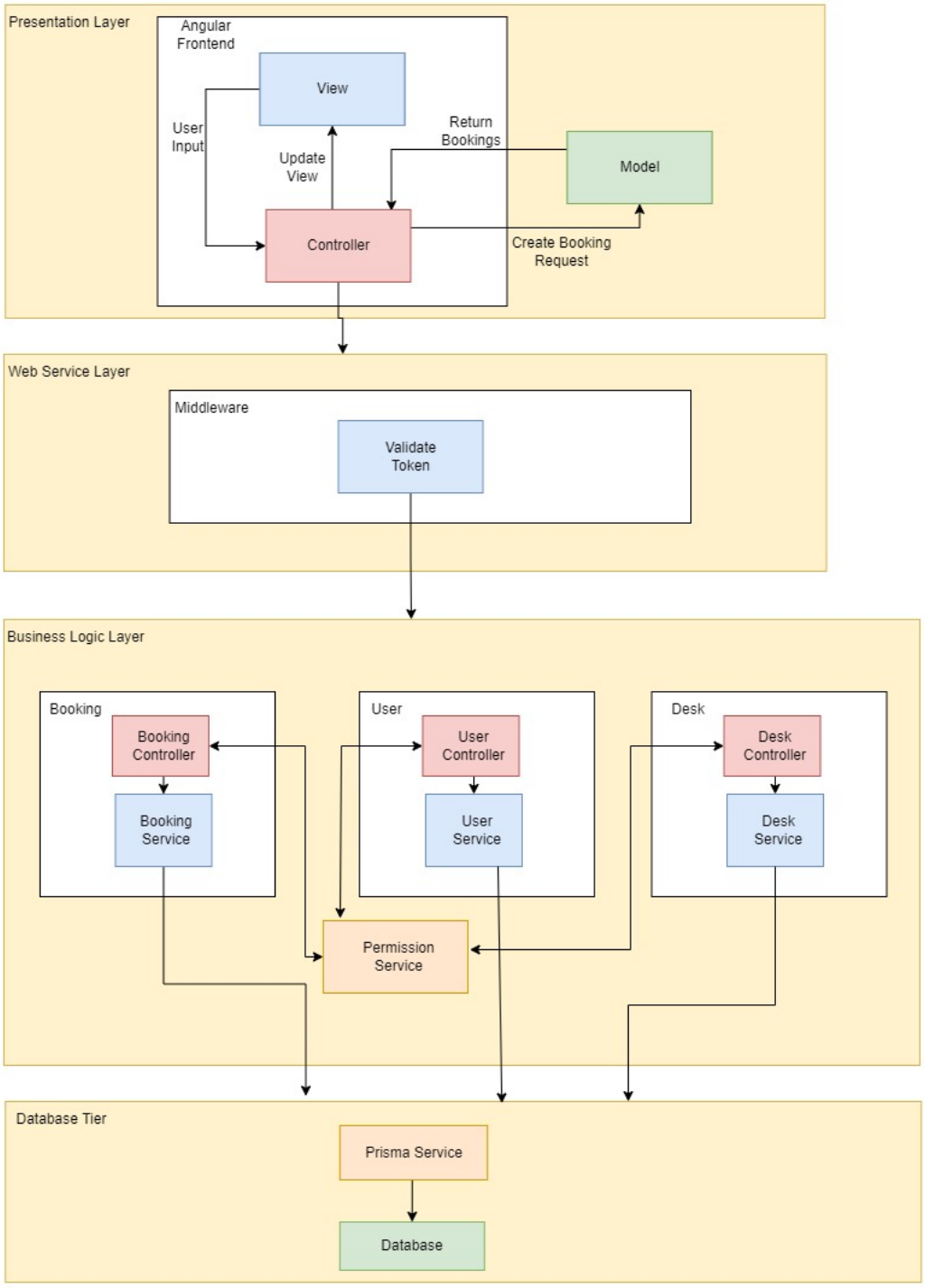

Figure 8: Architecture Pattern Diagram. MVC and Layered Patterns

8 Deployment Model

The Office Booker system is made up of three main components. These are the frontend, backend and database.

The hosting strategy uses docker containers for easy deployments. All docker containers mentioned are running on a free tier t3.micro (1 CPU and 512MB RAM) instance of AWS EC2 as specified by the system constraints.

The database is Postgresql and was chosen for ease of use. Version 14 of Postgresql is running.

The angular frontend got build to an html+css+js package, which is then hosted by nginx in a container. The frontend is accessible at https://officebooker.co.za

The nestjs backend is run inside a NodeJS LTS container and is hosted at https://api.officebooker.co.za/

For the purpose of the demo, Caddy is used as a reverse proxy to enable HTTPS and domain routing.

The backend container expects these environment variables:

- DATABASE_URL - A database url for prisma
  - `https://www.prisma.io/docs/reference/database-reference/connection-urls`
- COGNITO_USER_POOL_ID - Your AWS Cognito user pool id
- COGNITO_CLIENT_ID - Your AWS Cognito client id
- COGNITO_REGION - Your AWS Cognito region
- MAIL_HOST - SMTP Mail Host
- MAIL_USER - SMTP User
- MAIL_PASSWORD - SMTP Password
- MAIL_FROM - The Mail From Address
- MAIL_TRANSPORT - The URL
  - smtp://${MAIL_USER}:${MAIL_PASSWORD}@${MAIL_HOST}
  - filled with the above values.

Building the docker containers:

When running in production, the api url needs to specified. The api url is specified in apps/office-booker/src/environments/environment.prod.ts

First, we need nx to build our project:

Run while in the root of the project: yarn nx run-many –target=build –projects=api,office-booker –parallel

Then, to build the frontend: docker build -f ./apps/office-booker/Dockerfile . -t frontend

And the backend: docker build -f ./apps/api/Dockerfile . -t backend

Now the two containers are available on your system. A sample docker-compose is given.

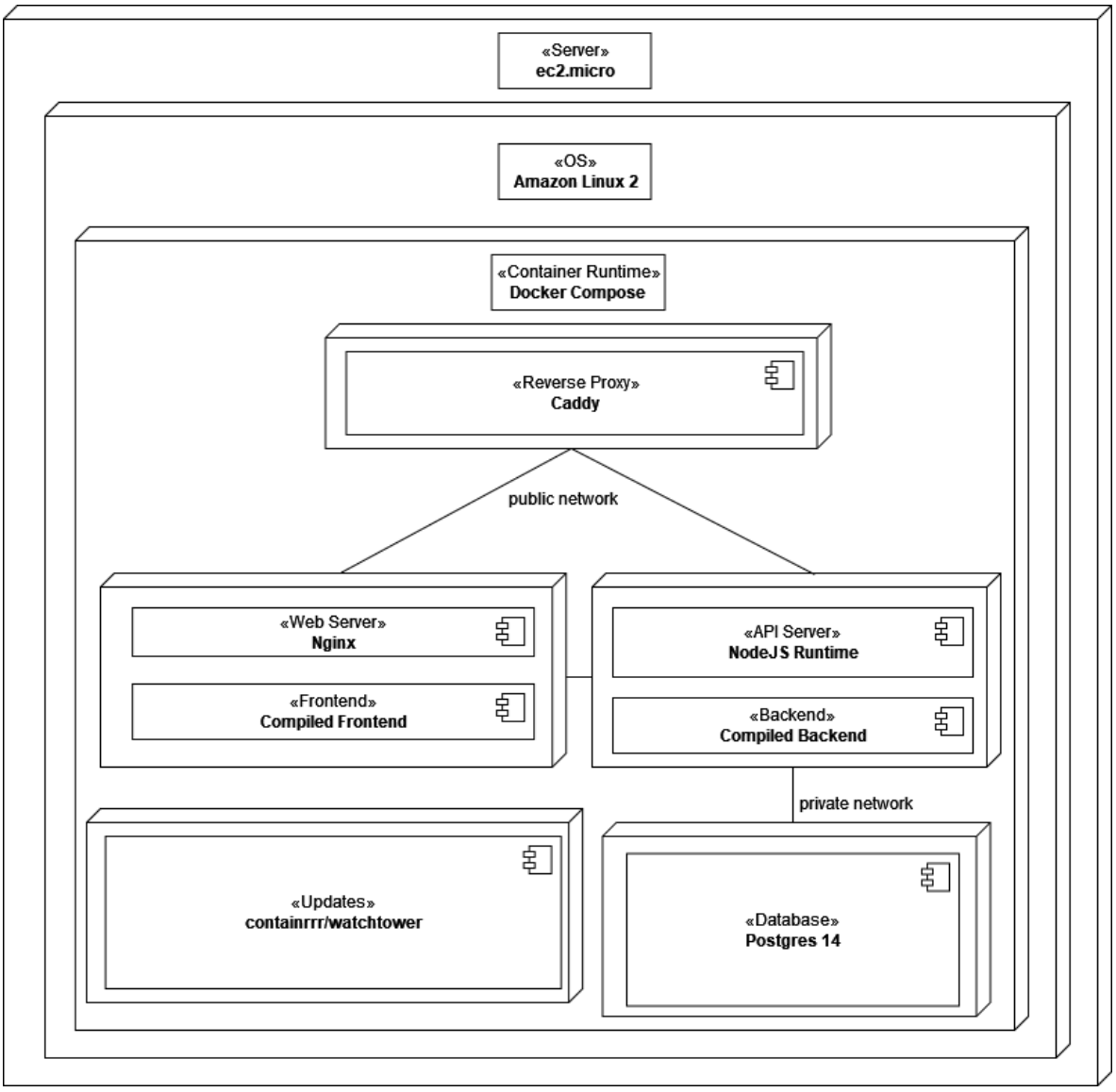

Figure 9: Simplified Deployment Diagram

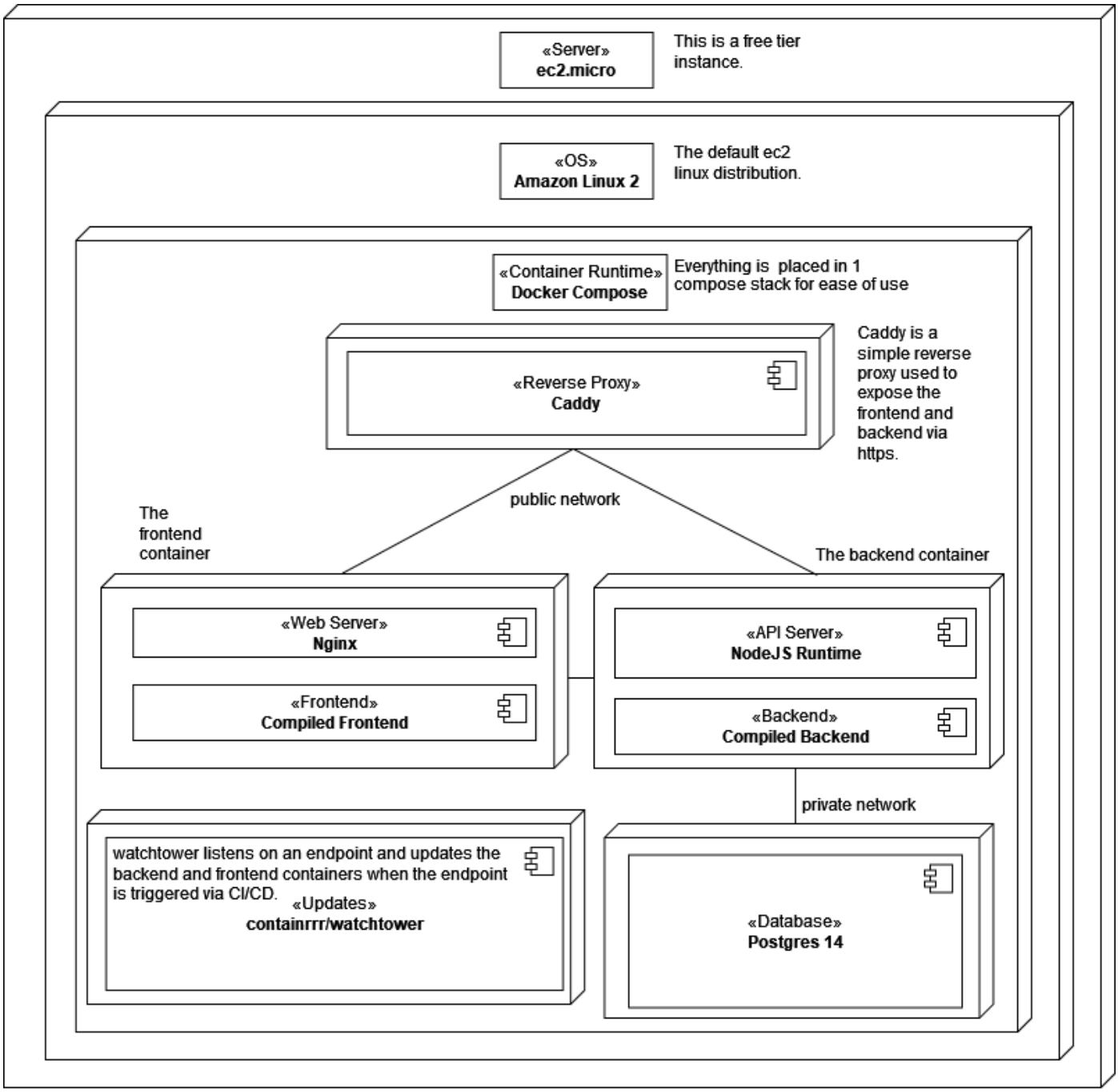

Figure 10: Detailed Deployment Diagram