A Unified Platform for Flight Management

GROUP 8

TOUFIC JRAB

VINCENT

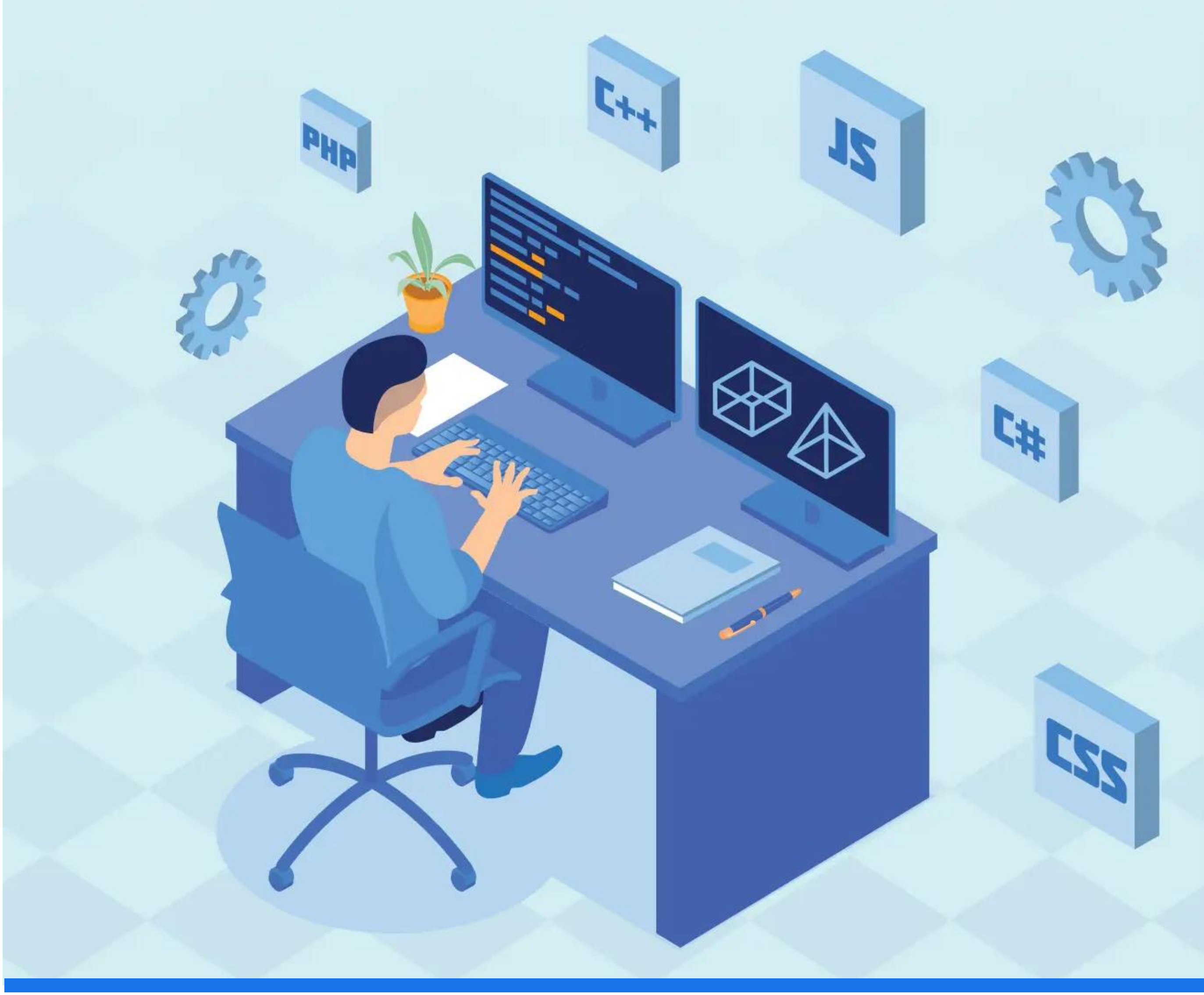
MARSHALL

ERIC

LINCE

The Problem

- The scope is to build a **web-based airline system** that allows:
- Customers to sign up, browse, and book flights (with 5% overbooking, rewards, cancellation/refunds).
- Pilots and flight attendants to view their assigned flights.
- Booking managers to create, edit, cancel flights and assign crew.
- The owner to manage employees, promotions/policies, and review statistics.



Some user/system requirements

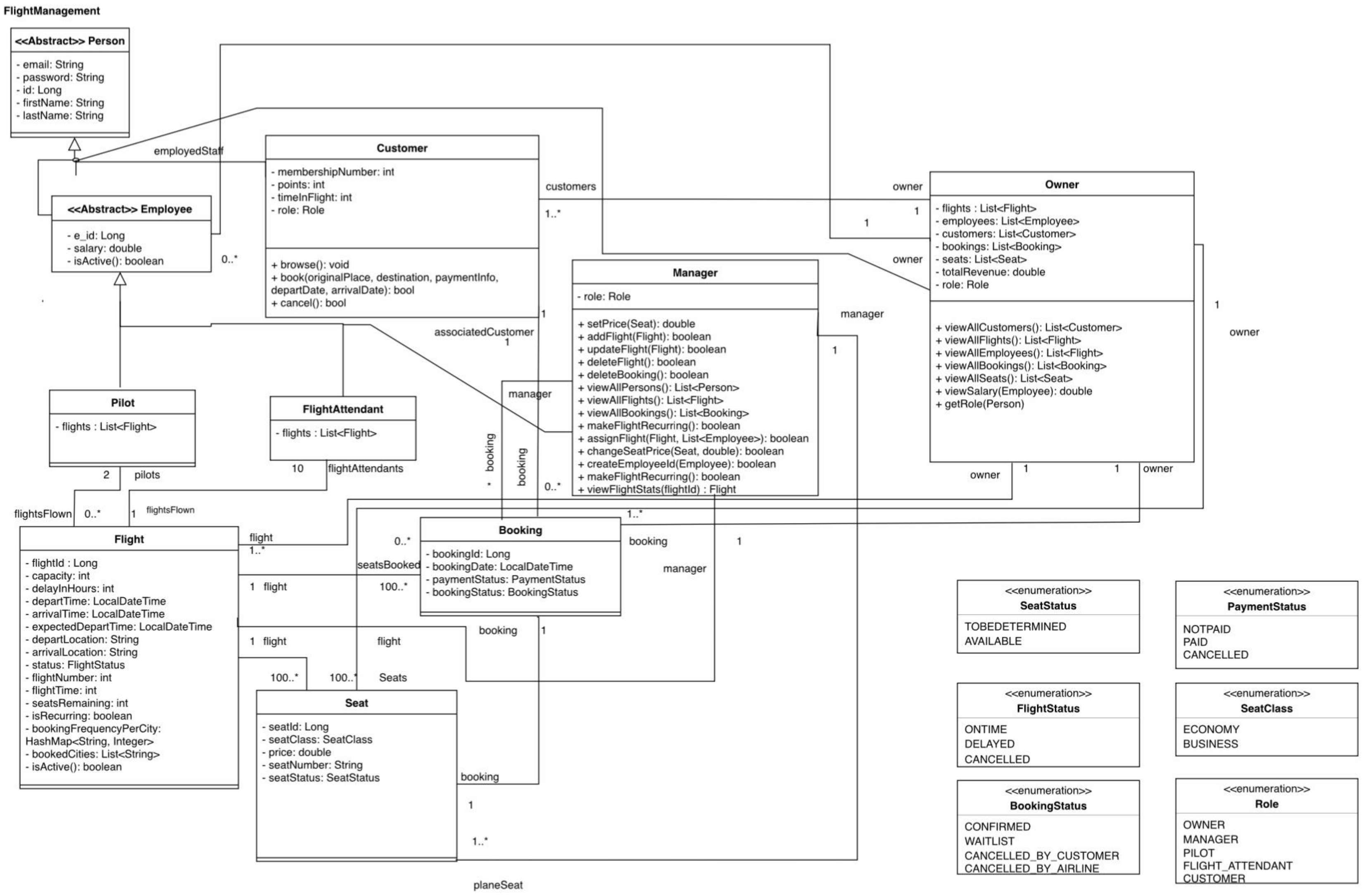
FUNCTIONAL

- The system shall prevent employees from booking using their employee accounts
- The system shall allow members to accumulate points for discounts
- The system shall ensure each customer can hold only one seat per flight, with optional seat selection; Economy and Business have distinct prices.
- Etc.

NON-FUNCTIONAL

- A customer shall be able to complete a booking in less than 5 clicks
- The system shall be easy to use and intuitive
- The system shall proceed transaction and redemption within 2 seconds after an order in 99 percent of cases
- The system shall restrict access to functions based on user roles
- Etc.

Our Solution



FlightManagement

<<Abstract>> Person
- email: String
- password: String
- id: Long
- firstName: String
- lastName: String

<<Abstract>> Employee
- e_id: Long
- salary: double
- isActive(): boolean

Customer
- membershipNumber: int
- points: int
- timeInFlight: int
- role: Role

+ browse(): void
+ book(originalPlace, destination, paymentInfo, departDate, arrivalDate): bool
+ cancel(): bool

Owner
- flights : List<Flight>
- employees: List<Employee>
- customers: List<Customer>
- bookings: List<Booking>
- seats: List<Seat>
- totalRevenue: double
- role: Role

+ viewAllCustomers(): List<Customer>
+ viewAllFlights(): List<Flight>
+ viewAllEmployees(): List<Flight>
+ viewAllBookings(): List<Booking>
+ viewAllSeats(): List<Seat>
+ viewSalary(Employee): double
+ getRole(Person)

Manager
- role: Role

+ setPrice(Seat): double
+ addFlight(Flight): boolean
+ updateFlight(Flight): boolean
+ deleteFlight(): boolean
+ deleteBooking(): boolean
+ viewAllPersons(): List<Person>
+ viewAllFlights(): List<Flight>
+ viewAllBookings(): List<Booking>
+ makeFlightRecurring(): boolean
+ assignFlight(Flight, List<Employee>): boolean
+ changeSeatPrice(Seat, double): boolean
+ createEmployeeId(Employee): boolean
+ makeFlightRecurring(): boolean
+ viewFlightStats(flightId) : Flight

Pilot
- flights : List<Flight>

FlightAttendant
- flights : List<Flight>

Booking
- bookingId: Long
- bookingDate: LocalDateTime
- paymentStatus: PaymentStatus
- bookingStatus: BookingStatus

Flight
- flightId : Long
- capacity: int
- delayInHours: int
- departTime: LocalDateTime
- arrivalTime: LocalDateTime
- expectedDepartTime: LocalDateTime
- departLocation: String
- arrivalLocation: String
- status: FlightStatus
- flightNumber: int
- flightTime: int
- seatsRemaining: int
- isRecurring: boolean
- bookingFrequencyPerCity: HashMap<String, Integer>
- bookedCities: List<String>
- isActive(): boolean

Seat
- seatId: Long
- seatClass: SeatClass
- price: double
- seatNumber: String
- seatStatus: SeatStatus

<<enumeration>>
SeatStatus
TOBEDETERMINED
AVAILABLE

<<enumeration>>
PaymentStatus
NOTPAID
PAID
CANCELLED

<<enumeration>>
FlightStatus
ONTIME
DELAYED
CANCELLED

<<enumeration>>
SeatClass
ECONOMY
BUSINESS

<<enumeration>>
BookingStatus
CONFIRMED
WAITLIST
CANCELLED_BY_CUSTOMER
CANCELLED_BY_AIRLINE

<<enumeration>>
Role
OWNER
MANAGER
PILOT
FLIGHT_ATTENDANT
CUSTOMER

Meet the Development Team

Vincent
Requirements & Documentation Lead

Fun fact: He likes playing cricket.

Eric
Backend & Testing Lead

Fun fact: He loves fishing

Marshall
UML & Architecture Lead

Fun fact: He hits 10k trophies in Clash Royale

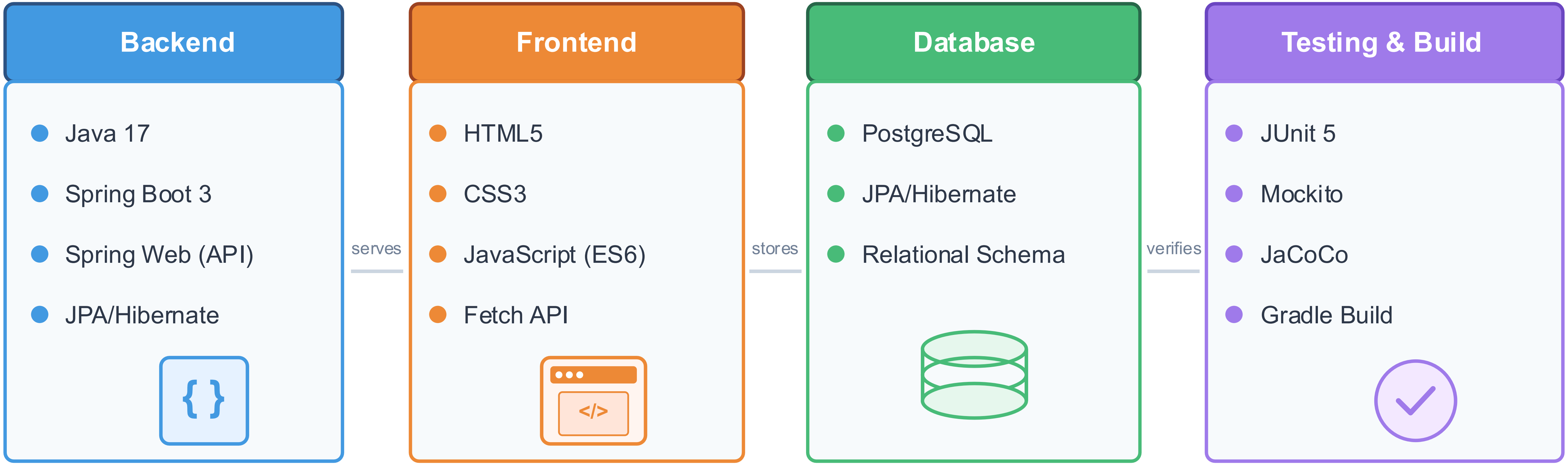
Lince
Database & Integration Lead

Fun fact: He is a cool dancer

Toufic
Project Manager and Scrum Master

Fun fact: He loves podcasts and hiking

Tech Stack: Built for Reliability & Growth



Backend

- Java 17
- Spring Boot 3
- Spring Web (API)
- JPA/Hibernate

serves

Frontend

- HTML5
- CSS3
- JavaScript (ES6)
- Fetch API

stores

Database

- PostgreSQL
- JPA/Hibernate
- Relational Schema

verifies

Testing & Build

- JUnit 5
- Mockito
- JaCoCo
- Gradle Build

All components work together in a unified Spring Boot application with comprehensive test coverage

Key Backend Features Implemented

Real airline operations powered by business logic

Flight & Seat Management

Flight Management

- Create/update/delete flights
- Enforce flight uniqueness
- Store times, dates, routes

Seat Management

- Seat maps with classes
- Pricing & availability status
- Prevent duplicate seat numbers

Overbooking Logic

- Allow up to **+5% overbooking**
- Controlled capacity rules

Customer & Bookings

Booking Logic

- Search by origin/destination/date
- Select seat → create booking
- Validate seat availability
- Prevent double-booking
- Cancel with status updates

Customer Handling

- Customer accounts
- Link customer ↔ bookings
- Retrieve upcoming bookings
- One seat per customer per flight

TICKET

User Types & Crew

User Type Management & Assignment

Customer Manager Owner

Pilot Flight Attendant

Crew Assignment

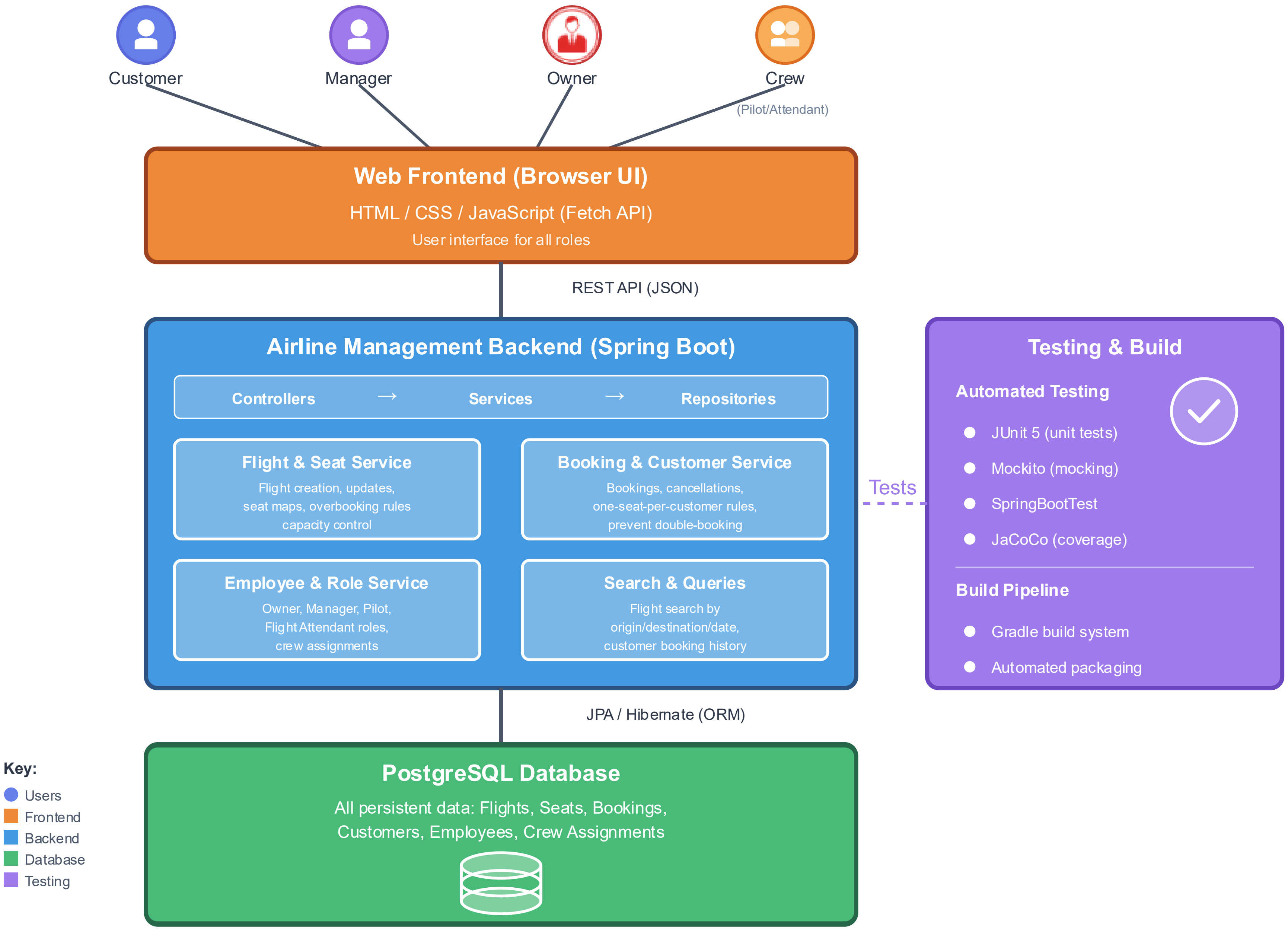
- Assign pilots to flights
- Assign attendants to flights
- Enforce crew requirements
- Crew view assigned flights

Owner Controls

- Oversee operations

Complete airline operations: Flight management • Customer bookings • Role-based access control

System Architecture: How Everything Fits Together



Customer

Manager

Owner

Crew

(Pilot/Attendant)

Web Frontend (Browser UI)

HTML / CSS / JavaScript (Fetch API)

User interface for all roles

REST API (JSON)

Airline Management Backend (Spring Boot)

Controllers → Services → Repositories

Flight & Seat Service

Flight creation, updates, seat maps, overbooking rules capacity control

Booking & Customer Service

Bookings, cancellations, one-seat-per-customer rules, prevent double-booking

Employee & Role Service

Owner, Manager, Pilot, Flight Attendant roles, crew assignments

Search & Queries

Flight search by origin/destination/date, customer booking history

Tests

Testing & Build

Automated Testing

- JUnit 5 (unit tests)
- Mockito (mocking)
- SpringBootTest
- JaCoCo (coverage)

Build Pipeline

- Gradle build system
- Automated packaging

JPA / Hibernate (ORM)

PostgreSQL Database

All persistent data: Flights, Seats, Bookings, Customers, Employees, Crew Assignments

Key:

● Users

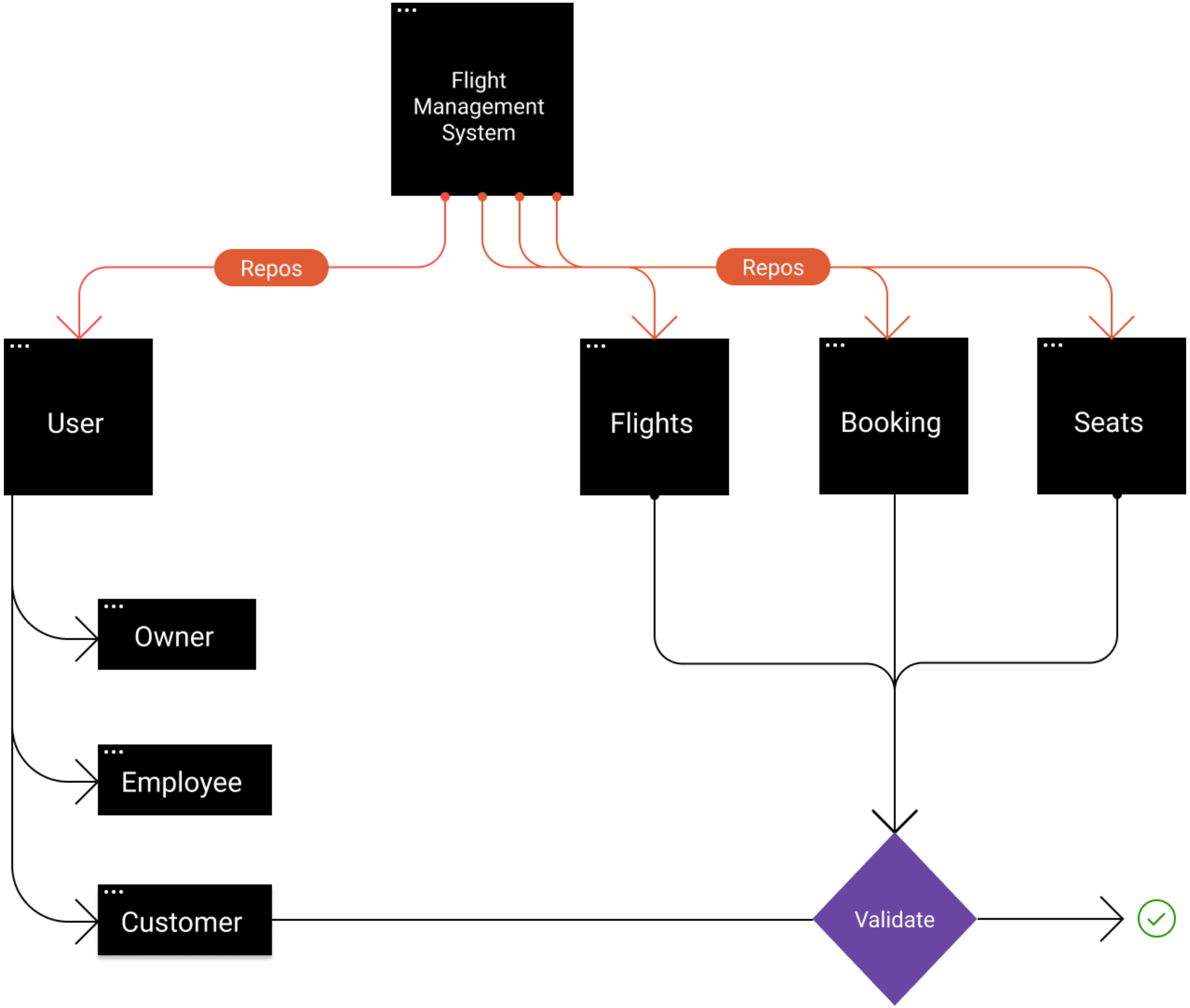
🟧 Frontend

🟦 Backend

🟩 Database

🟪 Testing

Flowchart

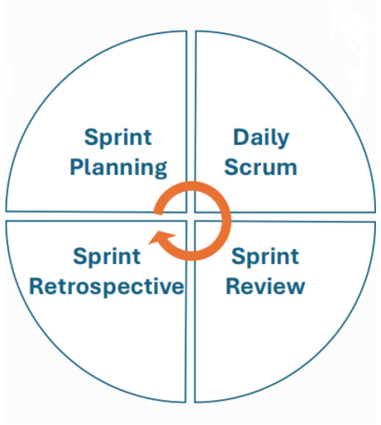

Flight Management System
Repos
Repos
User
Flights
Booking
Seats
Owner
Employee
Customer
Validate

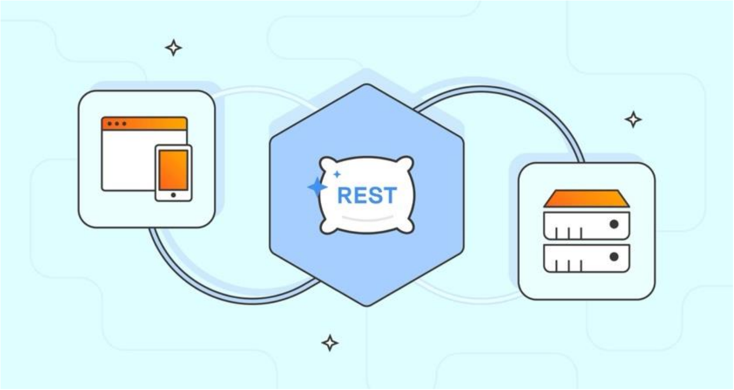
Demo

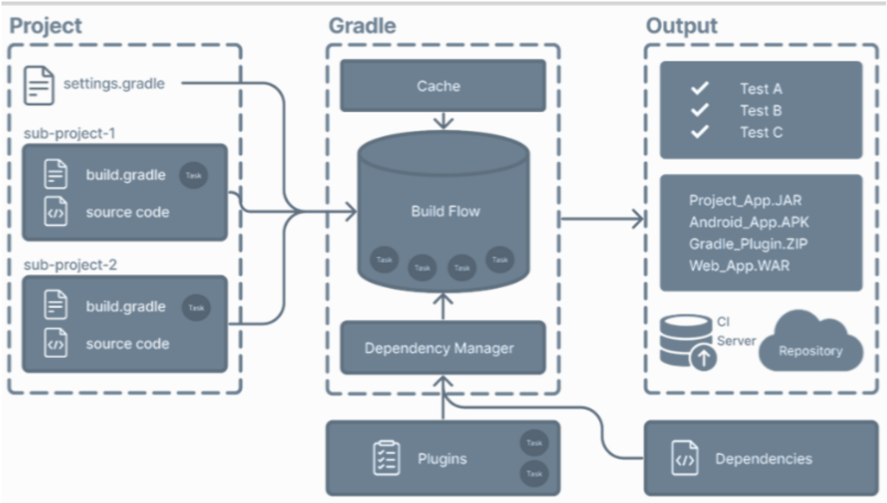
- Technical Demo: Features & Flow
- Business Demo: What this means to the owner as a business (Flow chart/business steps/input & output)

Software Development Approach

- Build System
  - Gradle 8 with Spring Boot 3.5 plugin
- RestAPI using Rest Controllers
- Agile using Scrum
  - Sprints are 1 week each
  - Releases are every Deliverable, so every 1 month
    - Oct: Domain model, Requirements and Database Design
    - Nov: Backend and Testing
    - Dec: Web Frontend

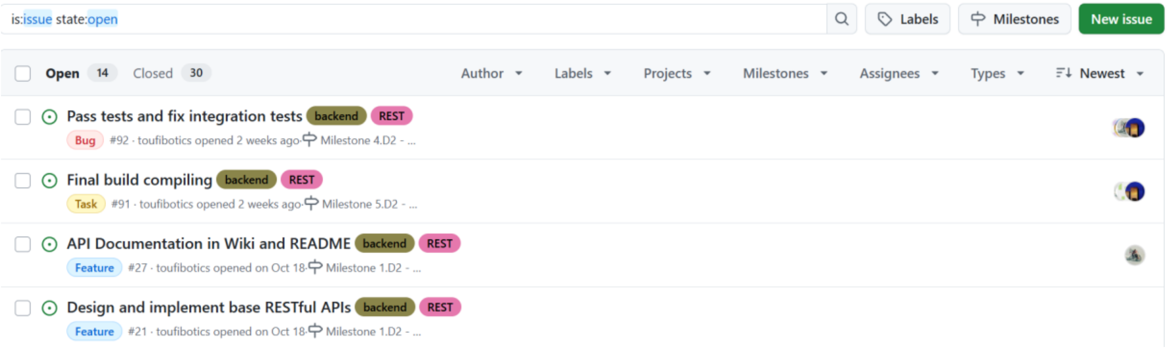






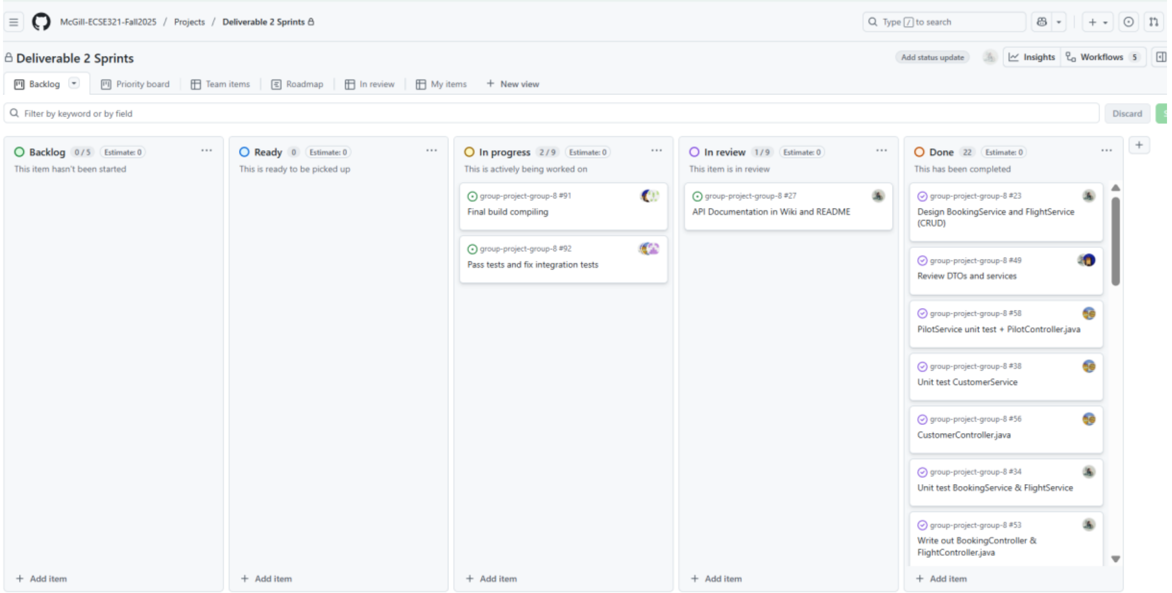
Software Development Approach

- GitHub
  - README with roles and effort tables
  - Wiki pages, where each is a Deliverable report
  - Projects board to track sprints and milestones
    - Backlog -> Ready -> In progress -> In review -> Done
  - Version Control
    - Feature branches and Pull Requests into Main
  - Issues to show attached assignee and milestone
  - Backlog to track sprints and milestones



Open 14 Closed 30

- Pass tests and fix integration tests backend REST
  Bug #92 · toufibotics opened 2 weeks ago · Milestone 4.D2 - ...
- Final build compiling backend REST
  Task #91 · toufibotics opened 2 weeks ago · Milestone 5.D2 - ...
- API Documentation in Wiki and README backend REST
  Feature #27 · toufibotics opened on Oct 18 · Milestone 1.D2 - ...
- Design and implement base RESTful APIs backend REST
  Feature #21 · toufibotics opened on Oct 18 · Milestone 1.D2 - ...



Deliverable 2 Sprints

Backlog 0 / 5 Estimate: 0
This item hasn't been started

Ready 0 Estimate: 0
This is ready to be picked up

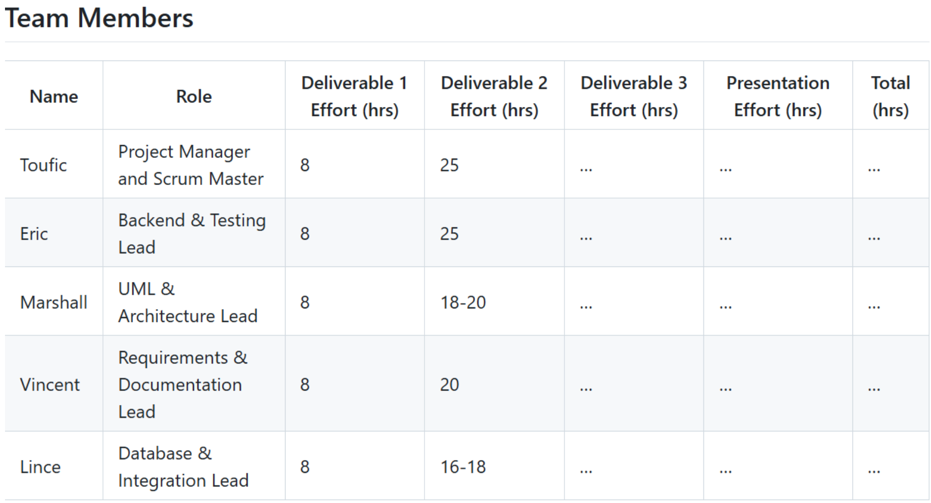
In progress 2 / 9 Estimate: 0
This is actively being worked on
- group-project-group-8 #91 Final build compiling
- group-project-group-8 #92 Pass tests and fix integration tests

In review 1 / 9 Estimate: 0
This item is in review
- group-project-group-8 #27 API Documentation in Wiki and README

Done 22 Estimate: 0
This has been completed
- group-project-group-8 #23 Design BookingService and FlightService (CRUD)
- group-project-group-8 #49 Review DTOs and services
- group-project-group-8 #58 PilotService unit test + PilotController.java
- group-project-group-8 #38 Unit test CustomerService
- group-project-group-8 #56 CustomerController.java
- group-project-group-8 #34 Unit test BookingService & FlightService
- group-project-group-8 #53 Write out BookingController & FlightController.java

Cost of Building & Releasing this System

- Total team hours spent on this project so far: 233 hours at $35 per hour: $8,155
  - Total Deliverable 1 Hours: 40
  - Total Deliverable 2 Hours: 108
  - Projected Total Deliverable 3 Hours: 85
- Releasing this system:
  - Web domain cost in Canada: $10.50
- Total system cost: $8,165.50



Team Members

Name	Role	Deliverable 1 Effort (hrs)	Deliverable 2 Effort (hrs)	Deliverable 3 Effort (hrs)	Presentation Effort (hrs)	Total (hrs)
Toufic	Project Manager and Scrum Master	8	25	...	...	...
Eric	Backend & Testing Lead	8	25	...	...	...
Marshall	UML & Architecture Lead	8	18-20	...	...	...
Vincent	Requirements & Documentation Lead	8	20	...	...	...
Lince	Database & Integration Lead	8	16-18	...	...	...

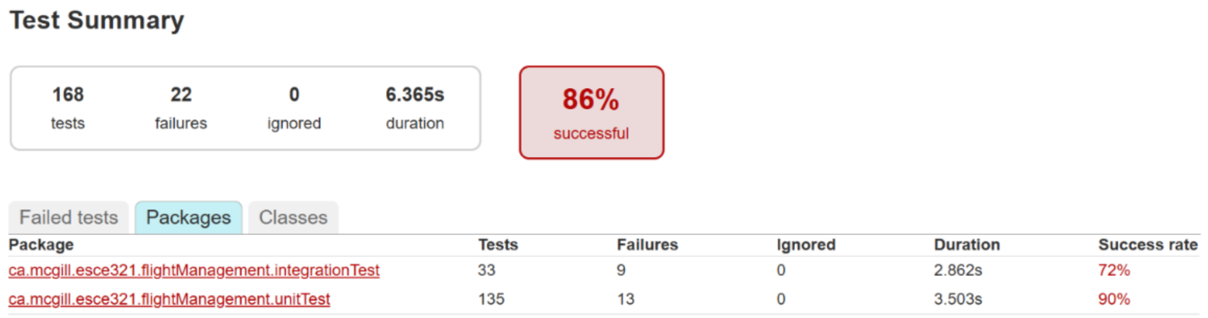
Why This System is Valuable

- For customers:
    - Complicated booking process made simple!

- For managers and owner:
    - Understandable dashboard shows all features

Tests & Compliance

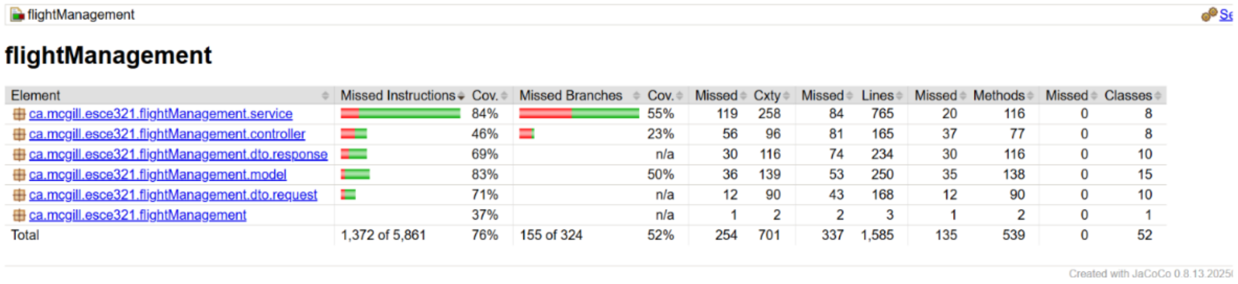
- Testing already done & branch coverage
- Compliance & Security: what are the codes &
  - Testing
    - JUnit 5
    - Mockito
    - JaCoCo plugin for test coverage



Test Summary

168	22	0	6.365s	86%
tests	failures	ignored	duration	successful

Package	Tests	Failures	Ignored	Duration	Success rate
ca.mcgill.esce321.flightManagement.integrationTest	33	9	0	2.862s	72%
ca.mcgill.esce321.flightManagement.unitTest	135	13	0	3.503s	90%

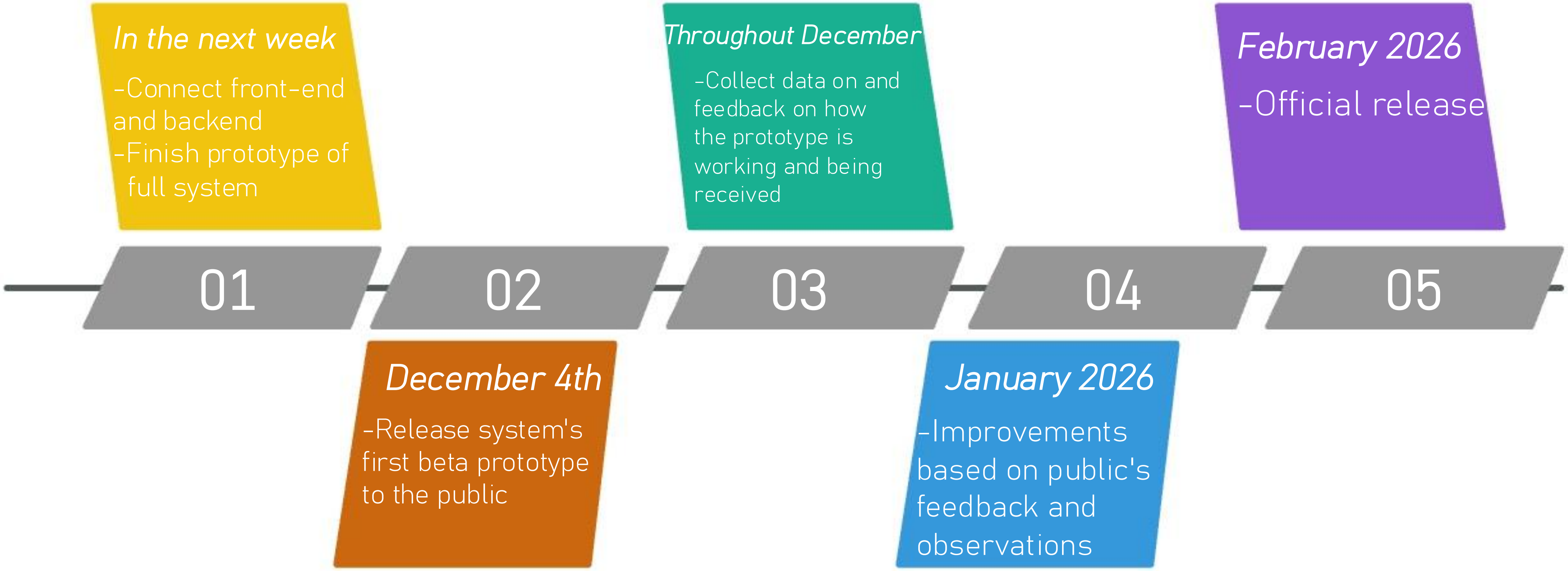


flightManagement

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
ca.mcgill.esce321.flightManagement.service	🟥🟩	84%	🟥🟩	55%	119	258	84	765	20	116	0	8
ca.mcgill.esce321.flightManagement.controller	🟥🟩	46%	🟥🟩	23%	56	96	81	165	37	77	0	8
ca.mcgill.esce321.flightManagement.dto.response	🟥🟩	69%		n/a	30	116	74	234	30	116	0	10
ca.mcgill.esce321.flightManagement.model	🟥🟩	83%		50%	36	139	53	250	35	138	0	15
ca.mcgill.esce321.flightManagement.dto.request	🟥🟩	71%		n/a	12	90	43	168	12	90	0	10
ca.mcgill.esce321.flightManagement		37%		n/a	1	2	2	3	1	2	0	1
Total	1,372 of 5,861	76%	155 of 324	52%	254	701	337	1,585	135	539	0	52

Created with JaCoCo 0.8.13.2025

Next Steps / Roadmap



01

In the next week

-Connect front-end and backend

-Finish prototype of full system

02

December 4th

-Release system's first beta prototype to the public

03

Throughout December

-Collect data on and feedback on how the prototype is working and being received

04

January 2026

-Improvements based on public's feedback and observations

05

February 2026

-Official release

Thank you for your time and welcome aboard!