REVISION HISTORY

Date	Version	Description	Author
21/03/24	1.0	2.2	Jeyhun Javadov
21/03/24	1.1	3.2	Ilkin Tomuev Rashad Musayev
21/03/24	2.0	4.2	Teymur Mammadov

TABLE OF CONTENTS

1 Introduction

1.1 Document overview

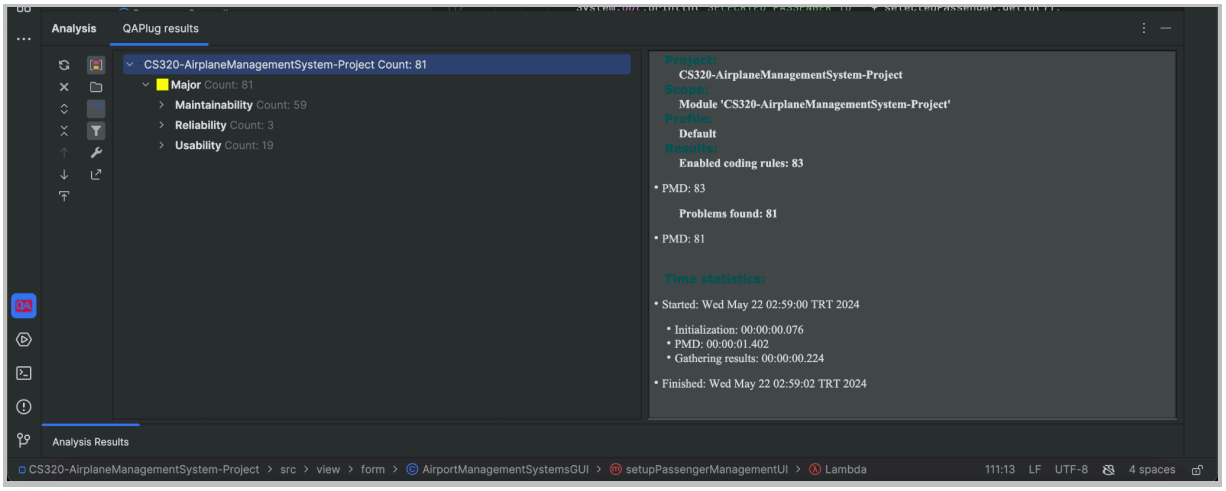
This document presents and interprets the code analysis results regarding the Airport Management System software development project. Code is analyzed by 3 different tools: 1) Static Code Analysis tool 2) Dependency Analysis tool and 3) Test Coverage tool. The first tool is used to reveal potential bugs that might be overseen during the testing process. The second tool is employed for evaluating the design quality based on the amount of coupling among the software modules and to what extent the code reflects the originally envisioned design. The last tool is used for measuring the coverage of unit tests in the project. Each section below is dedicated to each of these 3 analyses.

2 Static Code Analysis

2.1 Tools

We used PMD v6.21.0 (with QAPlug plugin) to do the analysis. It looks for bugs in Java code within Intellij IDEA.

2.2 Results and Discussion



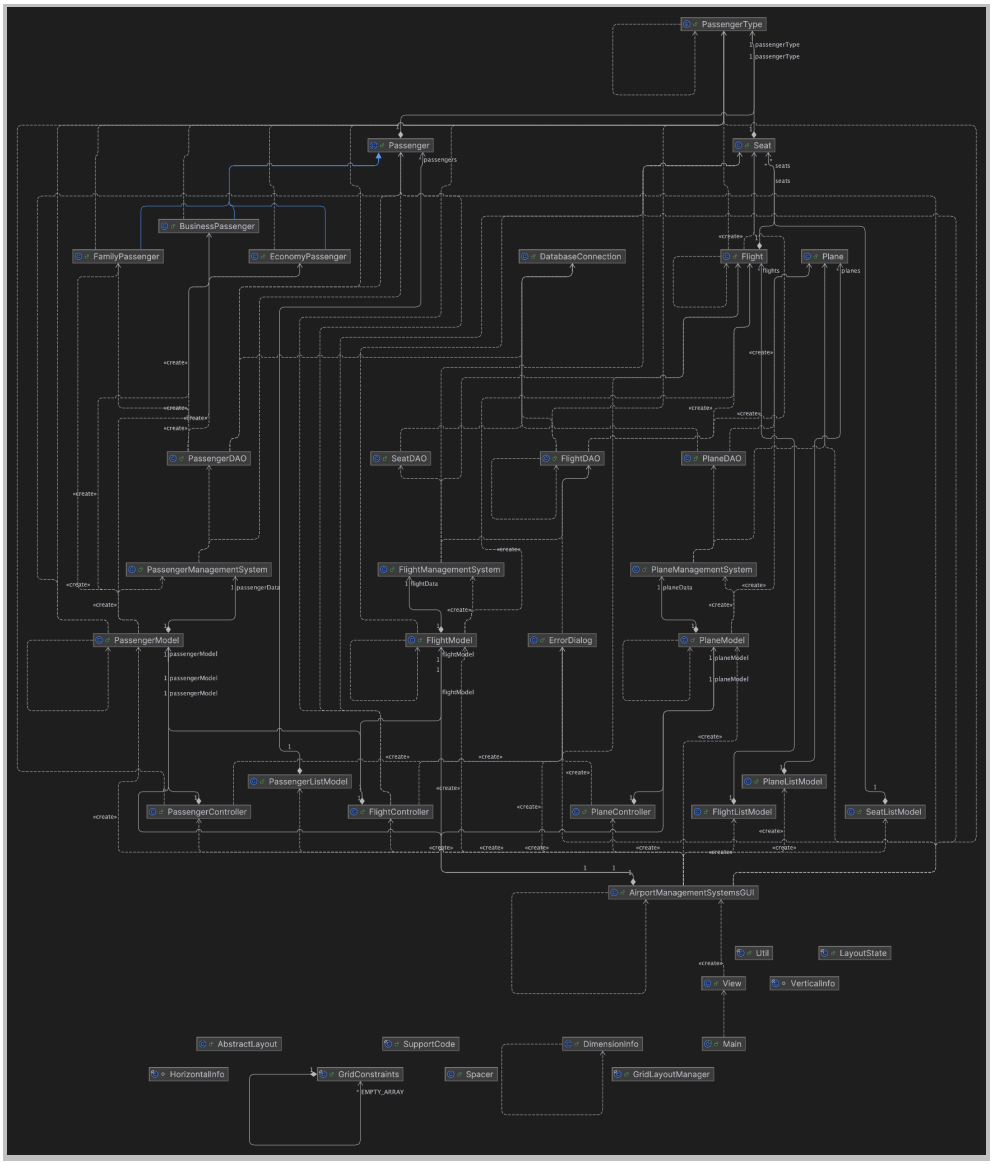
3 Dependency Analysis

3.1 Tools

IntelliJ IDEA Dependency Diagram tool is used to do dependency analysis in our project.

3.2 Results and Discussion

Dependency Graph for AMS is as follows



The extracted dependency graph is presented above. We calculated a set of metrics based on this graph as listed below:

# of Edges = 68
# of Nodes = 38
Edge-to-node Ratio = 62 / 38 = 1.63
Tree Impurity = 2 * (62 – 38 + 1) / (38 – 1) * (38 – 2) = 50 / 1332 ≈ 0.0375

The tree impurity proved to be relatively minor, as we have seen. It can indicate a well-designed construction. A balanced interconnection between classes is shown by the moderate edge-to-node ratio. On the other hand, since the majority of the edges are connected to these modules, we can observe that some classes, such as Passenger, PassengerDAO, Flight, Plane, and AirportManagementSystemGUI, are heavily linked. Refactoring and splitting these modules into distinct packages can lessen dense coupling at these individual modules.

4 Test Coverage Analysis

4.1 Tools

IntelliJ IDEA's Test Coverage built-in tool is used for test coverage analysis.

4.2 Results and Discussion

Element	Class, %	Method, %	Line, %	Branch, %
all	69% (16/23)	55% (61/110)	38% (141/370)	13% (26/192)
controller	0% (0/3)	0% (0/13)	0% (0/40)	0% (0/2)
FlightContro	0% (0/1)	0% (0/6)	0% (0/21)	0% (0/2)
PassengerC	0% (0/1)	0% (0/4)	0% (0/11)	100% (0/0)
PlaneContro	0% (0/1)	0% (0/3)	0% (0/8)	100% (0/0)
model	80% (16/20)	62% (61/97)	42% (141/330)	13% (26/190)
flight	88% (8/9)	57% (29/50)	46% (82/177)	18% (20/111)
data	80% (4/5)	50% (14/28)	40% (59/144)	14% (15/101)
Flight	100% (1/1)	61% (8/13)	82% (28/34)	68% (15/22)
FlightI	100% (1/1)	80% (4/5)	26% (22/82)	0% (0/66)
Seat	50% (1/2)	11% (1/9)	25% (4/16)	0% (0/3)
SeatD	100% (1/1)	100% (1/1)	41% (5/12)	0% (0/10)
list_mode	100% (2/2)	60% (6/10)	66% (8/12)	100% (0/0)
FlightI	100% (1/1)	60% (3/5)	66% (4/6)	100% (0/0)
SeatLi	100% (1/1)	60% (3/5)	66% (4/6)	100% (0/0)
FlightMar	100% (1/1)	80% (4/5)	80% (4/5)	100% (0/0)
FlightMo	100% (1/1)	71% (5/7)	68% (11/16)	50% (5/10)
passenger	75% (6/8)	74% (23/31)	43% (47/107)	12% (6/49)
passenger	75% (6/8)	74% (23/31)	43% (47/107)	12% (6/49)
data	71% (5/7)	76% (20/26)	42% (43/101)	12% (6/49)
Busine	100% (1/1)	100% (1/1)	100% (6/6)	100% (0/0)
Econo	100% (1/1)	100% (1/1)	100% (6/6)	100% (0/0)
Family	100% (1/1)	100% (1/1)	100% (6/6)	100% (0/0)
Passe	100% (1/1)	92% (12/13)	93% (14/15)	100% (2/2)
Passe	0% (0/2)	0% (0/5)	0% (0/57)	0% (0/43)
Passe	100% (1/1)	100% (5/5)	100% (11/11)	100% (4/4)
list_mode	100% (1/1)	60% (3/5)	66% (4/6)	100% (0/0)
Passe	100% (1/1)	60% (3/5)	66% (4/6)	100% (0/0)
plane	66% (2/3)	56% (9/16)	26% (12/46)	0% (0/30)
data	50% (1/2)	54% (6/11)	20% (8/40)	0% (0/30)
Plane	100% (1/1)	75% (6/8)	80% (8/10)	100% (0/0)
PlaneI	0% (0/1)	0% (0/3)	0% (0/30)	0% (0/30)
list_mode	100% (1/1)	60% (3/5)	66% (4/6)	100% (0/0)
Planel	100% (1/1)	60% (3/5)	66% (4/6)	100% (0/0)