FLIGHT BOOKING SYSTEM

No	Package	Class	Jumlah Method
1	Admin	ChangeFeatures.java	1
		SetSeats.java	1
2	Customer	BookFlight.java	1
		ChooseFlight.java	1
		CurrentBooking.java	2
		SearchFlight.java	1
3	Filters	SecurityFilter.java	3
		XSSFilter.java	3
		XSSRequestWrapper.java	5
4	Manager	ApproveFeatures.java	1
		ApproveSeats.java	1
		DissaproveFeatures.java	1
		DissaproveSeats.java	1
5	Models	Customer.java	4
		Employee.java	1
		FBS.java	6
		Features.java	35

		Flight.java	27
		Person.java	1
		Seat.java	4
6	Webservices	PriceAndSeats.java	2
7	<default package>	CustomerManager.java	1
		LoginManager.java	2
		LogoutManager.java	1
		OriginCompleter.java	1
		SContextListener.java	2
JUMLAH		26	109

1. Modularity

1.1. Coupling of Components Conformance

Nama	Coupling of Components Conformance
ID	MMo-1-G
Deskripsi	Seberapa independen sebuah komponen dan berapa banyak komponen yang bebas dari efek perubahan komponen lain dalam sistem
Rumus	X = A/B
A	Jumlah komponen yang diimplementasikan dengan sedikit pengaruh dengan komponen lain
B	Jumlah komponen yang harus independen
Perhitungan	Mengecek source code class file (selain Framework). Komponen yang memiliki dampak minimum adalah kelas yang tidak memerlukan perubahan dari kelas lain ketika kelas tersebut diubah. Komponen yang seharusnya independen adalah kelas yang berisi proses transaksional dengan basis data.

1.2. Cyclomatic Complexity

Nama	Cyclomatic Complexity
ID	MMo-2-S
Deskripsi	Jumlah modul yang mempunyai Cyclomatic Complexity dengan jumlah yang dapat diterima

Rumus	X = 1 -A / B
A	Jumlah modul yang dengan Cyclomatic Complexity melebihi batas
B	Jumlah seluruh modul
Perhitungan	Menghitung Cyclomatic Complexity padasetiap methodpada source code class file (selain Framework). Penghitungan dilakukan dengan bantuan aplikasi Cyvis[13]dengan threshold yaitu 10

1.3. Perhitungan

ID Modularity	**Perhitungan**			
	A	**B**	**Rumus**	**X**
MMo-1-G	14	20	X = A/B	0.7
MMo-2-S	0	109	X = 1 - A/B	1

1.4. Perhitungan Coupling of Component Conformance

No	Class	Coupling Class	A	B	X
1	ChangeFeatures.java	Features.java	1	1	1
2	SetSeats.java	Flight.java	1	1	1
3	BookFlight.java	-	0	0	0
4	CooseFlight.java	Customer.java	1	2	0.5
		Flight.java			
5	CurrentBooking.java	Customer.java	1	2	0.5
		Seat.java			
6	SearchFlight.java	FBS.java	1	3	0.33
		Features.java			
		Flight			
7	ApproveFeatures.java	Features.java	1	1	1
8	ApproveSeats.java	Flight.java	1	1	1
9	DissaproveFeatures.java	Features.java	1	1	1
10	DissaproveSeats.java	FBS.java	1	1	1
		Flight.java			
11	Customer.java	Person.java	1	1	1
12	Employee.java	Person.java	1	1	1
13	FBS.java	-	0	0	0
14	Features.java	-	0	0	0
15	Flight.java	-	0	0	0
16	Person.java	-	0	0	0
17	Seat.java	Features.java	1	1	1
18	CustomerManager.java	Customer.java	1	2	0.5
		FBS.java			
19	LoginManager.java	Customer.java	1	2	0.5
		FBS.java			
20	LogoutManager.java	-	0	0	0
Average					**0.5665**

1.5. Perhitungan Cyclomatic Complexity

No	Class	Method	Flow	E	N	G = E – N + 2

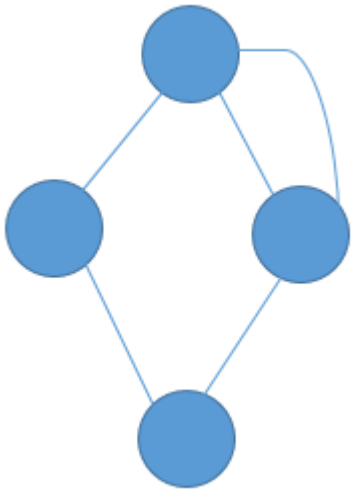
1.	ChangeFeatures.java	protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
		
		ArrayList<Features> f = (ArrayList<Features>) (getServletContext().getAttribute("features"));
		
		char[] s = {'e','b','f'};
		
		for (int i = 0; i < 3; i++)
		{
		//Saving old values
				2	2	2

		(f.get(i)).setNewSeatPitch(f.get(i).getSeatPitch()); (f.get(i)).setNewSeatWidth(f.get(i).getSeatWidth()); (f.get(i)).setNewVideoType(f.get(i).getVideoType()); (f.get(i)).setNewPowerType(f.get(i).getPowerType()); (f.get(i)).setNewSeatType(f.get(i).getSeatType()); (f.get(i)).setNewPrice(f.get(i).getPrice()); //Setting new values temporarily (f.get(i)).setSeatPitch((Double.parseDouble(request.getParameter("seat_pitch_" + s[i]))));				

		(f.get(i)).setSeatWidth((Double.parseDouble(request.getParameter("seat_width_" + s[i])))); (f.get(i)).setVideoType((request.getParameter("video_" + s[i]))); (f.get(i)).setPowerType((request.getParameter("power_" + s[i]))); (f.get(i)).setSeatType ((request.getParameter("seat_type_" + s[i]))); (f.get(i)).setPrice (Integer.parseInt(request.getParameter("price_" + s[i]))); } f.get(1).setNewWifi(f.get(1).getWifi());				

		f.get(2).setNewWifi(
		f.get(2).getWifi());
		
		f.get(1).setWifi(
		request.getParameter("wifi_b"));
		f.get(2).setWifi(
		request.getParameter("wifi_f"));
		
		f.get(2).setNewSpecialFood(f.get(2).
		getSpecialFood());
		f.get(2).setSpecialFood(
		request.getParameter("special_food_f"));
		
		Features.isChanged = true;				

		response.sendRedirect("ChangeFeat ures.jsp"); }				

2.	SetSeats.java	protected void doPost(HttpServletRequest
		request, HttpServletResponse response)
		throws ServletException,
		IOException {
		
		ArrayList<Flight> flights =
		(ArrayList<Flight>)
		(getServletContext().getAttribute("flights"));
		
		Flight flight = null;
		
		for (int i = 0; i < flights.size(); i++)
		{
				5	4	3

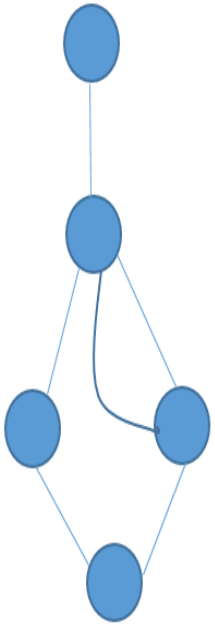
		if (flights.get(i).getFlightName().equals(request.getParameter("flight_name"))) { flight = flights.get(i); break; } } flight.setOldESeats(flight.getEconomySeats()); flight.setOldBSeats(flight.getBusinessSeats()); flight.setOldFSeats(flight.getFirstSeats());				

		flight.setOldTSeats(flight.getTotalSeats());				
		flight.setEconomySeats(Integer.parseInt (request.getParameter("seats_e")));				
		flight.setBusinessSeats(Integer.parseInt (request.getParameter("seats_b")));				
		flight.setFirstSeats(Integer.parseInt (request.getParameter("seats_f")));				
		flight.setTotalSeats(flight.getEconomySeats() + flight.getBusinessSeats() + flight.getFirstSeats());				

		flight.setCurrentSeats(flight.getTotalSeats());
		
		flight.isChanged = true;
		
		response.sendRedirect("SetSeats.jsp");
		}				

3.	BookFlight.java	protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {		1	2	1

		request.getRequestDispatcher("Book Flight.jsp").forward(request, response);
		
		}				

4.	ChooseFlight.java	protected void doPost(HttpServletRequest request, HttpServletResponse response)
		
		throws ServletException, IOException {
		
		ArrayList<Flight> flights = (ArrayList<Flight>) (getServletContext().getAttribute("flights"));
		
		Flight f = null;
		
		for (int i = 0; i < flights.size(); i++)
		
		{
				6	5	3

		if (flights.get(i).getFlightName().equals(request.getParameter("flight_name"))) { f = flights.get(i); break; } } f.setCustomer((Customer)(request.getSession().getAttribute("customer"))); request.getRequestDispatcher("CurrentBooking.do").forward(request, response); }				

5.	CurrentBooking.java	protected void doGet(HttpServletRequest request, HttpServletResponse response)
		throws ServletException, IOException {
		
		Customer customer = (Customer)(request.getSession(false).getAttribute("customer"));
		
		ArrayList<Seat> seats = customer.getCurrentBooking();
		
		request.setAttribute("seats", seats);
		
		request.getRequestDispatcher("CurrentBooking.jsp").forward(request,response);
		}
				1	2	1

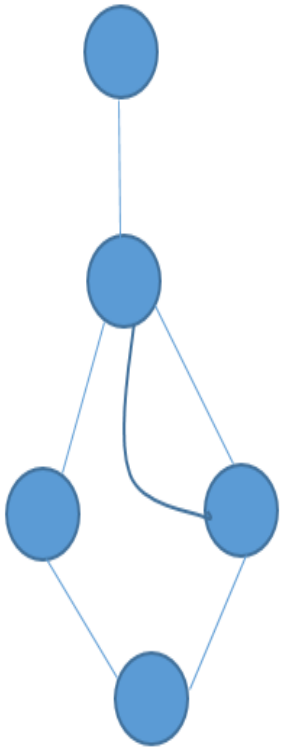
6.	SearchFlight.java	protected void doPost(HttpServletRequest request, HttpServletResponse response)
		throws ServletException, IOException {
		
		ArrayList<Flight> flights = (ArrayList<Flight>) (getServletContext().getAttribute("flights"));
		
		DateFormat df = new SimpleDateFormat("MM/dd/yyyy HH:mm:ss");
		
		ArrayList<Flight> results = new ArrayList();
		results.add(flights.get(1));
		request.setAttribute("results", results);
				1	2	1

		request.getRequestDispatcher("ShowFlights.jsp").forward(request,response);
		
		}				

7.	ApproveFeatures.java	protected void doPost(HttpServletRequest request, HttpServletResponse response)
		throws ServletException, IOException {
		
		ArrayList<Features> f = (ArrayList<Features>) (getServletContext().getAttribute("features"));
		
		for (int i = 0; i < 3; i++)
		{
				2	2	2

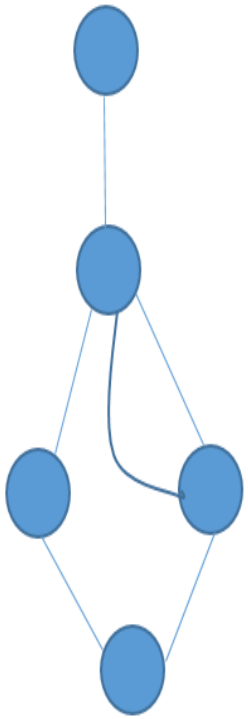
		(f.get(i)).setNewSeatPitch(0); (f.get(i)).setNewSeatWidth(0); (f.get(i)).setNewVideoType(null); (f.get(i)).setNewPowerType(null); (f.get(i)).setNewSeatType(null); (f.get(i)).setNewPrice(0); (f.get(i)).setNewWifi(null); (f.get(i)).setNewSpecialFood(null); } Features.isChanged = false;				

		response.sendRedirect("ApproveFeatures.jsp");
		}				

8.	ApproveSeats.java	protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException { ArrayList<Flight> flights = (ArrayList<Flight>) (getServletContext().getAttribute("flights")); Flight f = null; String a = request.getParameter("flight_name");		6	5	3

		for (int i = 0; i < flights.size(); i++)
		{
		if
		(flights.get(i).getFlightName().equals(request
		.getParameter("flight_name")))
		{
		f = flights.get(i);
		break;
		}
		}
		
		f.setOldESeats(0);
		f.setOldBSeats(0);
		f.setOldFSeats(0);
		f.setOldTSeats(0);
		
		f.isChanged = false;				

		response.sendRedirect("ApproveSeats.jsp");
		}				

9.	DissaproveFeatures.java	protected void doPost(HttpServletRequest request, HttpServletResponse response)
		throws ServletException, IOException {
		
		ArrayList<Features> f = (ArrayList<Features>) (getServletContext().getAttribute("features"));
		
		for (int i = 0; i < 3; i++)
		{
				6	5	3

		(f.get(i)).setSeatPitch(f.get(i).getNewSeatPitch()); (f.get(i)).setSeatWidth(f.get(i).getNewSeatWidth()); (f.get(i)).setVideoType(f.get(i).getNewVideoType()); (f.get(i)).setPowerType(f.get(i).getNewPowerType()); (f.get(i)).setSeatType(f.get(i).getNewSeatType()); (f.get(i)).setPrice(f.get(i).getNewPrice()); (f.get(i)).setWifi(f.get(i).getNewWifi()); (f.get(i)).setSpecialFood(f.get(i).getNewSpecialFood());				

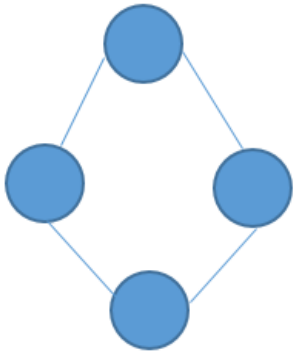
		(f.get(i)).setNewSeatPitch(0); (f.get(i)).setNewSeatWidth(0); (f.get(i)).setNewVideoType(null); (f.get(i)).setNewPowerType(null); (f.get(i)).setNewSeatType(null); (f.get(i)).setNewPrice(0); (f.get(i)).setNewWifi(null); (f.get(i)).setNewSpecialFood(null); } Features.isChanged = false;				

		response.sendRedirect("ApproveFeatures.jsp"); }				

10.	DissaproveSeats.java	protected void doPost(HttpServletRequest request, HttpServletResponse response)
		throws ServletException, IOException {
		
		ArrayList<Flight> flights = (ArrayList<Flight>) (getServletContext().getAttribute("flights"));
		
		Flight f = null;
		
		for (int i = 0; i < flights.size(); i++)
		{
				4	5	1

		if
		(flights.get(i).getFlightName().equals(request
		.getParameter("flight_name")))
		{
		f = flights.get(i);
		break;
		}
		}
		
		f.setEconomySeats(f.getOldESeats());
		f.setBusinessSeats(f.getOldBSeats());
		f.setFirstSeats(f.getOldFSeats());
		f.setTotalSeats(f.getOldTSeats());
		f.setCurrentSeats(f.getTotalSeats());
		
		f.setOldESeats(0);
		f.setOldBSeats(0);				

		f.setOldFSeats(0);
		f.setOldTSeats(0);
		
		f.isChanged = false;
		
		response.sendRedirect("ApproveSeats.jsp");
		}				

11.	Customer.java	public void setSeat(Seat s)
		{
		if(seats == null)
		seats = new ArrayList();
		
		seats.add(s);
		}
				4	4	2

12.	Employee.java	public Employee(String f, String e, String d)
		{
		super(f,e);
		employeeDesignation = d;
		}
				1	2	1

13.	FBS.java	public ArrayList<Customer> populateCustomers(Connection con) { ArrayList<Customer> customers = new ArrayList(); try { Statement stmt = con.createStatement(); String sql = "SELECT * FROM Customers";		2	2	2

		``ResultSet rs = stmt.executeQuery(sql);
		
		while (rs.next())
		{
		String name = rs.getString("name");
		String email = rs.getString("email");
		
		customers.add(new Customer(name,email, null));
		}
		}
		catch (SQLException ex) {
		
		Logger.getLogger(FBS.class.getName()).log(Level.SEVERE, null, ex);```				

		System.out.println(ex.getMessage()); } return customers; }				

14.	Features.java	public Features(int price, int newPrice, int t, boolean c, double sp, double sw, double newSp, double newSw, String vt, String newVt, String st, String newSt, String pt, String newPt, String w, String newW, String sf, String newSf)
		{
		this.price = price;
		this.newPrice = newPrice;
		type = t;
		isChanged = c;
				1	2	1

		seatPitch = sp;
		
		seatWidth = sw;
		
		newSeatPitch = newSp;
		
		newSeatWidth = newSw;
		
		videoType = vt;
		
		newVideoType = newVt;
		
		seatType = st;
		
		newSeatType = newSt;
		
		powerType = pt;
		
		newPowerType = newPt;
		
		wifi = w;
		
		newWifi = newW;				

		specialFood = sf;
		newSpecialFood = newSf;
		}				

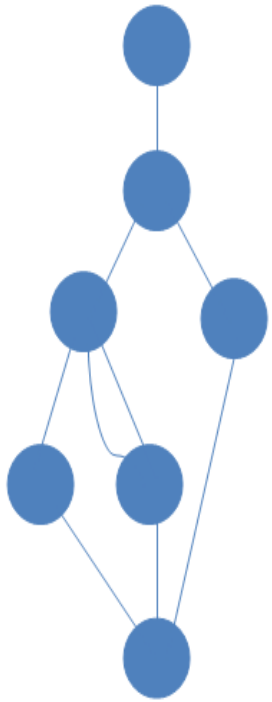
15.	Flight.java	public Flight(boolean c, int oldE, int oldB, int oldF, int oldT, String flightName, ArrayList<Seat> seats, int totalSeats, int currentSeats, String departureCity, String arrivalCity, Date departureDate, Date arrivalDate, int economySeats, int bSeats, int firstSeats)
		{
		isChanged = c;
		
		oldESeats = oldE;
		oldBSeats = oldB;
		oldFSeats = oldF;
		oldTSeats = oldT;
				1	2	1

		this.flightName = flightName;
		
		this.seats = seats;
		this.totalSeats = totalSeats;
		this.currentSeats = currentSeats;
		
		this.departureCity = departureCity;
		this.arrivalCity = arrivalCity;
		
		this.departureDate = departureDate;
		this.arrivalDate = arrivalDate;
		
		this.economySeats = economySeats;
		this.businessSeats = bSeats;
		this.firstSeats = firstSeats;				

		}				

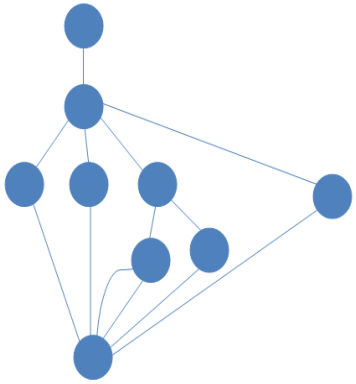
16.	Person.java	public Person(String f,String e)
		{
		name = f;
		email = e;
		}
				1	2	1

17.	Seat.java	public Seat(int sNumber, Flight flight, Features features, Customer c)
		{
		seatNumber = sNumber;
		this.flight = flight;
		this.features = features;
		this.c = c;
		}
				1	2	1

18.	CustomerManager.java	protected void doGet(HttpServletRequest request, HttpServletResponse response)
		throws ServletException, IOException {
		
		if (request.getSession().getAttribute("customer") == null){
		
		HttpSession s = request.getSession();
		String customerEmail = request.getRemoteUser();
		
		ArrayList<Customer> c = (ArrayList<Customer>)(getServletContext().getAttribute("customers"));
		
		for(int i = 0; i < c.size(); i++)
				9	7	4

		{
		if
		(c.get(i).getEmail().equals(customerEmail))
		{
		s.setAttribute("customer", c.get(i));
		
		break;
		}
		}
		
		}
		String uri = request.getRequestURI();
		String page = uri.split("/")[2];
		page = page.split(".jsp")[0] + ".jsp";				

		request.getRequestDispatcher(page) .forward(request,response);
		
		}				

19.	LoginManager.java	protected void doGet(HttpServletRequest request, HttpServletResponse response)
		
		throws ServletException, IOException {
		
		if(request.isUserInRole("Admin"))
		
		{
		
		response.sendRedirect("ChangeFeatures.jsp");
		
		}
		
		else if(request.isUserInRole("Manager"))
		
		{
				12	9	5

		response.sendRedirect("ApproveFeatures.jsp");
		}
		else if(request.isUserInRole("Customer"))
		{
		if (request.getSession().getAttribute("customer") == null){
		HttpSession s = request.getSession();
		String customerEmail = request.getRemoteUser();
		
		ArrayList<Customer> c = (ArrayList<Customer>)(getServletContext().getAttribute("customers"));				

		for(int i = 0; i < c.size(); i++)
		{
		if (c.get(i).getEmail().equals(customerEmail))
		{
		s.setAttribute("customer", c.get(i));
		break;
		}
		}
		}
		
		request.getRequestDispatcher("CurrentBooking.do").forward(request, response);
		
		}
		else				

		response.sendRedirect("home.jsp");
		
		}				

20.	LogoutManager.java	protected void doGet(HttpServletRequest request, HttpServletResponse response)
		
		throws ServletException, IOException {
		
		request.getSession().invalidate();
		
		response.sendRedirect("home.jsp");
		
		}
				1	2	1

2. Reusability

2.1. Reusability of Assets

Nama	Reusability of Assets
ID	MRe-1-G
Deskripsi	Jumlah aset yang bisa digunakan ulang

Rumus	X = A/B
A	Jumlah aset yang didesain dan diimplementasikan untuk dapat digunakan ulang
B	Jumlah seluruh asset
Perhitungan	Mengecek source code class file (selain Framework)·Komponen yang dibuat untuk dapat digunakan ulang adalah kelas yang berisi proses lebih dari satu fitur atau tidak dibuat spesifik untuk membantu satu kelas tertentu.

2.2. Conformance to Coding Rules

Nama	Conformance to Coding Rules
ID	MRe-2-S
Deskripsi	Jumlah modul yang dikembangkan sesuai coding rules
Rumus	X = A/B
A	Jumlah modul yang sesuai dengan coding rules
B	Jumlah modul dalam sistem
Perhitungan	Mengecek source code class file (selain Framework) Pengecekan dilakukan dengan bantuan sebuah tools yaitu Checkstyle, sebuah plugin dari Eclipse

2.3. Perhitungan

ID Reusability	Perhitungan			
	A	**B**	**Rumus**	**X**
MRe-1-G	5	26	X=A/B	0.1923
MRe-2_S	109	109	X=A/B	1

3. Analisability

3.1. System Log Completeness Conformance

Nama	System Log Completeness Conformance
ID	Man-1-G
Deskripsi	Seberapa jauh jangkauan log sistem dalam mencatat operasi-operasi dalam sistem
Rumus	X = A/B
A	Jumlah log yang direkam didalam sistem
B	Jumlah log dimana jejak prosesnya dibutuhkan sistem
Perhitungan	Mengecek adanya log di setiap fitur

3.2. Diagnosis Function Effectiveness

Nama	Diagnosis Function Effectiveness
ID	Man-2-S
Deskripsi	Proporsi sejauh mana implementasi fitur yang sesuai dengan kebutuhan dibandingkan dengan fitur yang telah diimplementasikan
Rumus	X = A/B
A	Jumlah fitur yang menjawab kebutuhan
B	Jumlah fitur yang ada pada program
Perhitungan	Mengecek proses utama dan dokumentasi

3.3. Diagnosis Function Sufficiency Conformance

Nama	Diagnosis Function Sufficiency Conformance
ID	Man-3-S
Deskripsi	Mengukur sejauh mana fitur-fitur memenuhi spesifikasi kebutuhan
Rumus	X = A/B
A	Jumlah fitur yang telah dibuat berdasarkan dokumentasi

B	Jumlah fitur yang seharusnya dibuat sesuai dengan dokumentasi
Perhitungan	Mengecek proses utama dan dokumentasi

3.4. Perhitungan

ID Analisability	**Perhitungan**			
	A	**B**	**Rumus**	**X**
MAn-1-G	0	26	X=A/B	0
MAn-2-S	10	10	X=A/B	1
Man-3-S	10	17	X=A/B	0.588

4. Testability

4.1. Test Function Completeness Conformance

Nama	Test Function Completeness Conformance
ID	MTe-1-G
Deskripsi	Seberapa lengkap uji coba terhadap sistem
Rumus	X = A/B
A	Jumlah uji coba yang dilakukan
B	Jumlah uji coba yang seharusnya dilakukan
Perhitungan	Mengecek proses utama dan dokumentasi

4.2. Autonomous Testability

Nama	Autonomous Testability
ID	MTe-2-S
Deskripsi	Seberapa kemampuan uji coba secara autonomous
Rumus	X = A/B

A	Jumlah stub yang dapat berjalan pada dependency test
B	Jumlah uji coba yang seharusnya dilakukan
Perhitungan	Mengecek proses utama dan dokumentasi

4.3. Test Restartability

Nama	Test Restartability
ID	MTe-3-S
Deskripsi	Seberapa mudah uji coba ulang setelah melakukan perbaikan sistem
Rumus	X = A/B
A	Jumlah uji coba yang memiliki titik pause dan restart
B	Jumlah fitur yang seharusnya dibuat sesuai dengan dokumentasi
Perhitungan	Mengecek proses utama dan dokumentasi

4.4. Perhitungan

ID Testability	Perhitungan			
	A	B	Rumus	X
MTe-1-G	17	17	X=A/B	1
MTe-2-S	0	17	X=A/B	0
MTe-3-S	0	17	X=A/B	0