**Software Design Description**

**for**

**Java Air**

**Prepared by Rui Zhang**

**Team: Avian Limited**

# Introduction

## Purpose

The Software Requirements Specification (SRS) is intended to describe the software architecture and design in detail, which is used to guide the team members’ code development. This document also describe the implantation of front-end and back-end in detail.

## Scope

This document will cover the software design for release version 0.0.1 of the Java Air software product, and the functional and non-functional requirements in the Software Requirements Specification (SRS) will be satisfied. The details of architecture, module description, process description, database design and algorithm design will be demonstrated.

## Definitions, acronyms, and abbreviations

CI = Configuration Item

CMMI = Capability Maturity Model Integration

IEEE = Institute of Electrical and Electronics Engineers

QA = Quality Assurance

SEI = Software Engineering Institute

SCMP = Software Configuration Management Plan

SPMP = Software Project Management Plan (this document)

SRS = Software Requirements Specification

SDD = Software Design Document

SQAP = Software Quality Assurance Plan

SVVP = Software Verification and Validation Plan

STP = Software Test Plan

UD = User Documentation

WBS = Work Breakdown Structure

U/PD = User/Product Director

PM = Project Manager

RE = Requirement Engineer

SA = Software Architect

IE = Integration Engineer

TE = Testing Engineer

CD = Code Developer

# Reference

Software Engineering Modern Approaches, 2nd ed. By Eric J. Braude, Michael E. Bernstein.

Software Configuration Management Plan (SCMP) for Java Air version 1.0

Software Design Description (SDD) for Java Air version 1.0

Software Project Management Plan (SPMP) for Java Air version 1.0

Software Quality Assurance Plan (SQAP) for Java Air version 1.0

Software User Documentation Plan (SUDP) for Java Air version 1.0

Software Test Document (STD) for Java Air version 1.0

Software Verification and Validation Plan (SVVP) for Java Air version 1.0

# Decomposition description

## Module decomposition

### CustomerAccount Entity

This entity holds all customer account for user of Java Air Company. A detailed description will be given in section 6.

### Flight Entity

This entity is used to hold all Flight information of Java Air Company. A detailed description will be given in section 6.

### ReservationBuilder Entity

This is an entity consisting of reservation information. A detailed description will be given in section 6.

### Employee Entity

This is the list of all employee. A detailed description will be given in section 6.

### Passenger Entity

This entity holds all passenger information for Java Air Flight. A detailed description will be given in section 6.

### Aircraft Entity

This simply entity holds different kinds of aircraft information of Java Air Company. A detailed description will be given in section 6.

## 3.2 Concurrent process decomposition

### 3.2.1 Ticket Builder Entity

This entity is used to create a reservation, based on the user input of passenger information and flight information. This entity can be considered one of the basic entity in this system.

### 3.2.2 Order Reservation Entity

This entity is intended to store reservation information into database file.

### 3.2.3 Order Customer Account Entity

This entity is used to store Customer information into database file.

### 3.2.4 Order Flight Entity

This entity is used to store Customer information into database file.

## 3.3 Data decomposition

### 3.3.1 Customer Account Entity

This entity represents the customer account, which holds all related qualities about customer, such as name, gender, data of birth and so on. A detailed description will be given in section 6.

### 3.3.2 Flight Entity

This entity holds the all qualities of flights. A detailed description will be given in section 6.

### 3.3.3 Reservation Entity

The reservation qualities are held in this entity, like reservation number, passenger information, flight number and so on. A detailed description will be given in section 6.

### 3.3.4 Employee Entity

This entity represents the detailed account information of employee. A detailed description will be given in section 6.

# 4. Dependency description

## 4.1 Intermodule dependencies

## 4.2 Interprocess dependencies

## 4.3 Data dependencies

# 5. Interface description

## 5.1 Module interface

### 5.1.1 Java Air Home Interface

### 5.1.2 Customer Account Registration

### 5.1.3 Customer Account Management: Welcome

### 5.1.4 Customer Account Management: Update Personal Information

### 5.1.5 Customer Account Management: View Existing Reservations

### 5.1.6 Customer Account Management: View Rewards Program Information

### 5.1.7 Customer Account Management: Reset Password

### 5.1.8 Flight Reservation: Search Flights

### 5.1.9 Flight Reservation: Purchase Flights

## 5.2 Process interface

### 5.2.1 Customer account landing interface

# 6. Detailed design

## 6.1 Module detailed design

### 6.1.1 CustomerAccount Detail

CustomerAccount is a module for holding the data of all customer and transferring data to database module. The purpose of this module to be generated is to store the related customer information from database and allocate all customer interface applicants.

The attributes in this module are shown below:

* emailAddress: This attribute holds the customer email address. For unregister customer, this attribute will be set as null.
* customerFirstName: This attribute holds the customer first name. The name information for unregister customer will be obtained from reservation step.
* customerMidName: This attribute holds the customer mid name. The name information for unregister customer will be obtained from reservation step.
* customerLastName: This attribute holds the customer last name. The name information for unregister customer will be obtained from reservation step.
* customerTelephNum: This attribute holds the customer telephone number. The telephone number information for unregister customer will be obtained from reservation step.
* customerPassword: This attribute holds the customer password. The password for unregister customer will be a default string.
* customerID: This attribute holds the customer ID number, which is specific for every customer.
* customerDataBirth: This attribute holds the customer data birth in string type, and the day, month and year are separated by comma. The name information for unregister customer will be obtained from reservation step.
* customerGender: This attribute holds the customer gender. The gender information for unregister customer will be obtained from reservation step.
* currentRewardPoint: This attribute holds the customer current reward point.
* totalRewardPoint: This attribute holds the customer total reward point on history, and this function could be used to identify whether a customer is new one.
* paymentInform: This attribute holds the customer payment information, including the payment method, card number, expire data, card holder name and security code in string type.

There functions in this module are shown below:

* CustomerAccount(): This is constructor for this module.
* generCustomID(): This function will be used to generate a specific customer ID
* setCustomerID(in customerID:String): The customer ID will be set in CustomerAccount Object by this function.
* getCustomerID(): This function will return the customer account ID in string type.
* nameSeperate(in customerName:String): This fuction will separate the whole name string into first name, mid name and last name, the names are stored at local variables.
* nameMerge(): This fuction will merge parts of name and return in whole name string.
* setCustomerFirstName(in customerFirstName:String): The customer first name will be set in CustomerAccount Object by this function.
* getCustomerFirstName(): This function will return the customer first name in string type.
* setCustomerMidName(in customerFirstName:String): The customer mid name will be set in CustomerAccount Object by this function.
* getCustomerMidName(): This function will return the customer mid name in string type.
* setCustomerLastName(in customerFirstName:String): The customer last name will be set in CustomerAccount Object by this function.
* getCustomerLastName(): This function will return the customer last name in string type.
* setCustomerTelephNum(in customerTelehpNum:int): The customer telephone number will be set in CustomerAccount Object by this function.
* getCustomerTelephNum(): This function will return the customer telephone number in string type.
* setCustomerPassword(in customerPassword:String): The customer password will be set in CustomerAccount Object by this function.
* getCustomerPassword(): This function will return the customer password in string type.
* setCustomerGender(in customerGender:String): The customer gender will be set in CustomerAccount Object by this function.
* getCustomerGender(): This function will return the customer gender in string type. setCustomerNowReward(in reward:int): The customer current rewards will be added or be decreased in CustomerAccount Object by this function.
* getCustomerNowReward(): This function will return the customer current reward in string type.
* setCustomerTotalReward(in totalReward:int): The customer total rewards will be added in CustomerAccount Object by this function.
* getCustomerTotalReward: This function will return the customer total reward in string type.
* setCustomerDataBirth(in dataBirth:String): The customer data birth will be set in CustomerAccount Object by this function.
* getCustomerDataBirth(): This function will return the customer data birth reward in string type.

### 6.1.2 Flight Detail

Flight is a module for holding the flight information from database and transferring data to ReservationBuilder module. The purpose of this module to support the search flight interface applicants.

The attributes in this module are shown below:

* flightNum: This attribute holds the origin airport name.
* originAirport: This attribute holds the origin airport name.
* destinationAirport: This attribute holds the destination airport name.
* transferAirport: This attribute holds the transfer airport name.
* scheduledDepartureData: This attribute holds the scheduled departure data.
* scheduledArrivalData: This attribute holds the scheduled arrival data.
* scheduledDepartureTime: This attribute holds the scheduled departure time.
* scheduledArrivalTime: This attribute holds the scheduled arrival time.
* actualDepartureData: This attribute holds the actual departure data.
* actualArrivalData: This attribute holds the actual arrival data.
* actualDepartureTime: This attribute holds the actual departure time.
* actualArrivalTime: This attribute holds the actual arrival time.
* prepareTime: This attribute holds the required preparation time for depature.
* aircraftType: This attribute holds the aircraft type of flight.
* tripMileage: This attribute holds the trip mileage.

There functions in this module are shown below:

* Flight(): This is constructor for this module.
* generFlightNum(): This function will be used to generate a flight number
* setFlightNum(in flightNum:String): The flight number will be set in Flight object by this function.
* getFlightNum (): This function will return the flight number in string type.
* setOriginAirport(in originAirport:String): The origin airport name will be set in Flight object by this function.
* getOriginAirport (): This function will return the origin airport name in string type.
* setTransferAirport(in transferAirport:String): The transfer airport name will be set in Flight object by this function.
* getTransferAirport (): This function will return the transfer airport name in string type.
* setDestinationAirport(in destinationAirport:String): The destinationairport name will be set in Flight object by this function.
* getDestinationAirport (): This function will return the destinationairport name in string type.
* setScheduledDepartureData(in scheduledDepartureData:String): The scheduled departure data will be set in Flight object by this function.
* getScheduledDepartureData(): This function will return the scheduled departure data in string type.
* setScheduledArrivalData(in scheduledArrivalData:String): The scheduled arrival data will be set in Flight object by this function.
* getScheduledArrivalData(): This function will return the scheduled arrival data in string type.
* setScheduledDepartureTime(in scheduledDepartureTime:String): The scheduled departure data will be set in Flight object by this function.
* getScheduledDepartureTime (): This function will return the scheduled departure data in string type
* setScheduledArrivalTime (in scheduledArrivalTime:String): The scheduled arrival time will be set in Flight object by this function.
* getScheduledArrivalTime (): This function will return the scheduled arrival time in string type.
* setActualDepartureData(in actualDepartureData:String): The actual departure data will be set in Flight object by this function.
* getActualDepartureData(): This function will return the actual departure data in string type.
* setActualArrivalData(in actualArrivalData:String): The actual arrival data will be set in Flight object by this function.
* getActualArrivalData(): This function will return the actual arrival data in string type.
* setActualDepartureTime(in actualDepartureTime:String): The actual departure data will be set in Flight object by this function.
* getActualDepartureTime (): This function will return the actual departure data in string type
* setActualArrivalTime (in actualArrivalTime:String): The actual arrival time will be set in Flight object by this function.
* getActualArrivalTime (): This function will return the actual arrival time in string type.
* setPrepareTime (in prepareTime:String): The preparation time for departure will be increased or decreased in Flight object by this function.
* getPrepareTime(): This function will return the preparation time for departure in string type.
* setAircraftType (in aircraftType:String): The aircraft type will be set in Flight object by this function.
* getAircraftType (): This function will return the aircraft type in string type.
* setTripMileage (in tripMileage:String): The trip mileage will be set in Flight object by this function.
* getAircraftType (): This function will return the trip mileage in string type.

### 6.1.3 ReservationBuilder Detail

ReservationBuilder is a module for holding the reservation information from database and transferring data to user interface module. The purpose of this module to support the reservation applicants.

The attributes in this module are shown below:

* customerAccount: This attribute holds the customer account reference.
* price: This attribute holds the price number.
* paymentInform: This attribute holds the payment information.
* billAddress: This attribute holds the bill address information.
* travelType: This attribute holds the travel type.
* reservationNum: This attribute holds the reservation number.
* passengerList: This attribute holds the passenger object list.
* primaryPassenger: This attribute holds the primary passenger reference.
* departureFlight: This attribute holds the departure flight reference.
* returnFlight: This attribute holds the return flight reference.
* actualArrivalTime: This attribute holds the actual arrival time.

There functions in this module are shown below:

* TicketBuilder(): This is constructor for this module.
* TicketBuilder(in customerAccount:RegisterCustomerAccount): This is constructor for this module, which is used for register customer interface.
* generateReservationNum(): String: This function will be used to generate a reservation number
* setReservationNum(in flightNum:String): The reservatioin number will be set in Flight object by this function.
* getReservationNum (): This function will return the reservation number in string type.

### 6.1.4 Employee Detail

### 6.1.5 Passenger Detail

### 6.1.6 Aircraft Detail

## 6.2 Data detailed design

### 6.2.1 Data entity 1 detail

### 6.2.2 Data entity 2 detail