FLYAWAY (AN AIRLINE BOOKING PORTAL).

[WRITE UP]

Sameer Khan

PROJECT: **FlyAway (An Airline Booking Portal).**

(Write up)

Submitted by:- **Sameer Khan**

GitHub repository link**:**
https://github.com/SameerKhan0411/Flyaway.git

INDEX

SPRINTS PLANNING:

The project is planned to be completed in 2 sprints.

Tasks assumed to be completed in the sprint 1 are:

- Creating the flow of the application.
- Initializing git repository to track changes as development progresses.
- Creating the database in MySQL.

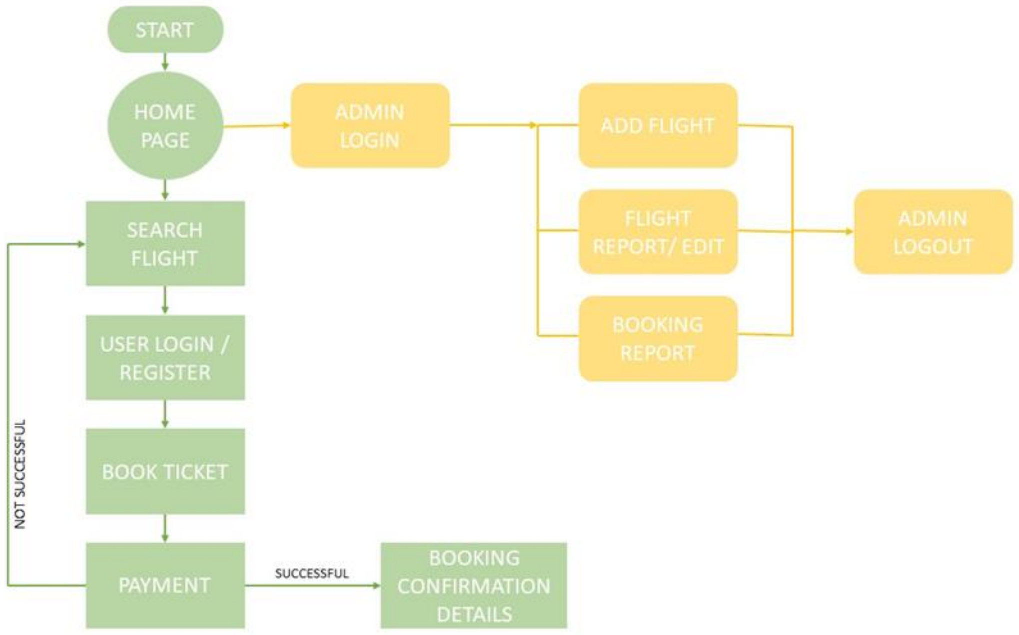
Tasks assumed to be completed in the sprint 2 are:

- Creating the Dynamic Web Project and importing the required dependencies.
- Writing the code & testing the code.
- Pushing code to GitHub.
- Creating this specification document highlighting application capabilities, appearance, and user interactions.

Concepts used in project:

- **Eclipse** is an integrated development environment used in computer programming. It contains a base workspace and an extensible plug-in system for customizing the environment.
- **Java** is a high-level, class-based, object-oriented programming language that is designed to have as few implementation dependencies as possible.
- **SQL** is a domain-specific language used in programming and designed for managing data held in a relational database management system, or for stream processing in a relational data stream management system.
- **Maven** is a build automation tool used primarily for Java projects. Maven can also be used to build and manage projects written in C#, Ruby, Scala, and other languages. The Maven project is hosted by the Apache Software Foundation, where it was formerly part of the Jakarta Project.
- **Scrum** is a framework for developing, delivering, and sustaining products in a complex environment, with an initial emphasis on software development, although it has been used in other fields including research, sales, marketing and advanced technologies.

Flow Chart/ Flow diagram of program.



Development steps of program.

BookModel.java

```
package in.co.air.line.ticket.model;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;

import org.apache.log4j.Logger;

import in.co.air.line.ticket.bean.BookBean;
import in.co.air.line.ticket.bean.FlightBean;
import in.co.air.line.ticket.exception.ApplicationException;
import in.co.air.line.ticket.exception.DatabaseException;
import in.co.air.line.ticket.exception.DuplicateRecordException;
import in.co.air.line.ticket.util.JDBCDataSource;

public class BookModel {

	private static Logger log = Logger.getLogger(BookModel.class);

	public Integer nextPK() throws DatabaseException {
		log.debug("Model nextPK Started");
		Connection conn = null;
		int pk = 0;

		try {
			conn = JDBCDataSource.getConnection();
			PreparedStatement pstmt = conn.prepareStatement("SELECT MAX(ID)
FROM A_Book");
			ResultSet rs = pstmt.executeQuery();
			while (rs.next()) {
				pk = rs.getInt(1);
			}
			rs.close();

		} catch (Exception e) {
			log.error("Database Exception..", e);
			throw new DatabaseException("Exception : Exception in getting
PK");
		} finally {
			JDBCDataSource.closeConnection(conn);
		}
		log.debug("Model nextPK End");
		return pk + 1;
	}
	public long add(BookBean bean) throws ApplicationException,
DuplicateRecordException {

		Connection conn = null;
		int pk = 0;

		FlightModel fModel = new FlightModel();
		FlightBean fBean = fModel.findByPK(bean.getFlightId());
		// bean.setFinalPrice((fBean.getTicketPrice()*bean.getNoOfPerson()));

		try {
			conn = JDBCDataSource.getConnection();
			pk = nextPK();
			// Get auto-generated next primary key
			System.out.println(pk + " in ModelJDBC");
			conn.setAutoCommit(false); // Begin transaction
			PreparedStatement pstmt = conn
					.prepareStatement("INSERT INTO A_Book
VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)");
			pstmt.setInt(1, pk);
			pstmt.setLong(2, bean.getFlightId());
			pstmt.setString(3, fBean.getFightName());
			pstmt.setString(4, bean.getFirstName());
			pstmt.setString(5, bean.getLastName());
			pstmt.setString(6, bean.getMobileNo());
			pstmt.setDate(7, new
java.sql.Date(bean.getBookDate().getTime()));
			pstmt.setString(8, bean.getEmailId());
			pstmt.setString(9, bean.getAddress());
			pstmt.setLong(10, bean.getNoOfPerson());
			pstmt.setLong(11, fBean.getTicketPrice());
			pstmt.setLong(12, bean.getFinalPrice());
			pstmt.setString(13, bean.getCreatedBy());
			pstmt.setString(14, bean.getModifiedBy());
			pstmt.setTimestamp(15, bean.getCreatedDatetime());
			pstmt.setTimestamp(16, bean.getModifiedDatetime());
			pstmt.executeUpdate();
			conn.commit(); // End transaction
			pstmt.close();
		} catch (Exception e) {

			try {
				conn.rollback();
			} catch (Exception ex) {
				ex.printStackTrace();
				throw new ApplicationException("Exception : add rollback
exception " + ex.getMessage());
			}
			throw new ApplicationException("Exception : Exception in add
User");
		} finally {
			JDBCDataSource.closeConnection(conn);
		}

		return pk;
	}

	public BookBean findByName(String name) throws ApplicationException {
		log.debug("Model findByLogin Started");
		StringBuffer sql = new StringBuffer("SELECT * FROM A_Book WHERE 
firstName=?");
		BookBean bean = null;
		Connection conn = null;
		System.out.println("sql" + sql);

		try {
			conn = JDBCDataSource.getConnection();
			PreparedStatement pstmt = conn.prepareStatement(sql.toString());
			pstmt.setString(1, name);
			ResultSet rs = pstmt.executeQuery();
			while (rs.next()) {
				bean = new BookBean();
				bean.setId(rs.getLong(1));
				bean.setFlightId(rs.getLong(2));
				bean.setFlightName(rs.getString(3));
				bean.setFirstName(rs.getString(4));
				bean.setLastName(rs.getString(5));
				bean.setMobileNo(rs.getString(6));
				bean.setBookDate(rs.getDate(7));
				bean.setEmailId(rs.getString(8));
				bean.setAddress(rs.getString(9));
				bean.setNoOfPerson(rs.getLong(10));
				bean.setPrice(rs.getLong(11));
				bean.setFinalPrice(rs.getLong(12));
				bean.setCreatedBy(rs.getString(13));
				bean.setModifiedBy(rs.getString(14));
				bean.setCreatedDatetime(rs.getTimestamp(15));
				bean.setModifiedDatetime(rs.getTimestamp(16));

			}
			rs.close();
		} catch (Exception e) {
			e.printStackTrace();
			log.error("Database Exception..", e);
			throw new ApplicationException("Exception : Exception in getting 
User by login");
		} finally {
			JDBCDataSource.closeConnection(conn);
		}
		log.debug("Model findByLogin End");
		return bean;
	}

	public BookBean findByPK(long pk) throws ApplicationException {
		log.debug("Model findByPK Started");
		StringBuffer sql = new StringBuffer("SELECT * FROM A_Book WHERE ID=?");
		BookBean bean = null;
		Connection conn = null;

		try {
					conn = JDBCDataSource.getConnection();
					PreparedStatement pstmt = conn.prepareStatement(sql.toString());
					pstmt.setLong(1, pk);
					ResultSet rs = pstmt.executeQuery();
					while (rs.next()) {
						bean = new BookBean();
						bean.setId(rs.getLong(1));
						bean.setFlightId(rs.getLong(2));
						bean.setFlightName(rs.getString(3));
						bean.setFirstName(rs.getString(4));
						bean.setLastName(rs.getString(5));
						bean.setMobileNo(rs.getString(6));
						bean.setBookDate(rs.getDate(7));
						bean.setEmailId(rs.getString(8));
						bean.setAddress(rs.getString(9));
						bean.setNoOfPerson(rs.getLong(10));
						bean.setPrice(rs.getLong(11));
						bean.setFinalPrice(rs.getLong(12));
						bean.setCreatedBy(rs.getString(13));
						bean.setModifiedBy(rs.getString(14));
						bean.setCreatedDatetime(rs.getTimestamp(15));
						bean.setModifiedDatetime(rs.getTimestamp(16));

					}
					rs.close();
				} catch (Exception e) {
					e.printStackTrace();
					log.error("Database Exception..", e);
					throw new ApplicationException("Exception : Exception in getting
User by pk");
				} finally {
					JDBCDataSource.closeConnection(conn);
				}
				log.debug("Model findByPK End");
				return bean;
			}

			public void delete(BookBean bean) throws ApplicationException {

				Connection conn = null;
				try {
					conn = JDBCDataSource.getConnection();
					conn.setAutoCommit(false); // Begin transaction
					PreparedStatement pstmt = conn.prepareStatement("DELETE FROM
A_Book WHERE ID=?");
					pstmt.setLong(1, bean.getId());
					pstmt.executeUpdate();
					conn.commit(); // End transaction
					pstmt.close();

				} catch (Exception e) {

					try {
						conn.rollback();
					} catch (Exception ex) {
```

```
                                throw new ApplicationException("Exception : Delete
rollback exception " + ex.getMessage());
                        }
                        throw new ApplicationException("Exception : Exception in delete
User");
                } finally {
                        JDBCDataSource.closeConnection(conn);
                }

        }

        public void update(BookBean bean) throws ApplicationException,
DuplicateRecordException {
                log.debug("Model update Started");
                Connection conn = null;

                FlightModel fModel = new FlightModel();
                FlightBean fBean = fModel.findByPK(bean.getFlightId());
                bean.setFinalPrice((fBean.getTicketPrice() * bean.getNoOfPerson()));

                try {
                        conn = JDBCDataSource.getConnection();
                        conn.setAutoCommit(false); // Begin transaction
                        PreparedStatement pstmt = conn.prepareStatement(
                                        "UPDATE A_Book SET
FlightId=?,FlightName=?,FirstName=?,LastName=?,MobileNo=?,BookDate=?,EmailId=?,Addres
s=?,NoOfPerson=?,Price=?,FinalPrice=?,"
                                                        +
"CREATEDBY=?,MODIFIEDBY=?,CREATEDDATETIME=?,MODIFIEDDATETIME=? WHERE ID=?");
                        pstmt.setLong(1, bean.getFlightId());
                        pstmt.setString(2, fBean.getFightName());
                        pstmt.setString(3, bean.getFirstName());
                        pstmt.setString(4, bean.getLastName());
                        pstmt.setString(5, bean.getMobileNo());
                        pstmt.setDate(6, new
java.sql.Date(bean.getBookDate().getTime()));
                        pstmt.setString(7, bean.getEmailId());
                        pstmt.setString(8, bean.getAddress());
                        pstmt.setLong(9, bean.getNoOfPerson());
                        pstmt.setLong(10, fBean.getTicketPrice());
                        pstmt.setLong(11, bean.getFinalPrice());
                        pstmt.setString(12, bean.getCreatedBy());
                        pstmt.setString(13, bean.getModifiedBy());
                        pstmt.setTimestamp(14, bean.getCreatedDatetime());
                        pstmt.setTimestamp(15, bean.getModifiedDatetime());
                        pstmt.setLong(16, bean.getId());
                        pstmt.executeUpdate();
                        conn.commit(); // End transaction
                        pstmt.close();
                } catch (Exception e) {
                        e.printStackTrace();
                        log.error("Database Exception..", e);
                        try {
                                conn.rollback();
                        } catch (Exception ex) {
```

```
                        throw new ApplicationException("Exception : Delete
rollback exception " + ex.getMessage());
                    }
                    throw new ApplicationException("Exception in updating User ");
            } finally {
                    JDBCDataSource.closeConnection(conn);
            }
            log.debug("Model update End");
    }

    public List search(BookBean bean) throws ApplicationException {
            return search(bean, 0, 0);
    }

    public List search(BookBean bean, int pageNo, int pageSize) throws
ApplicationException {
            log.debug("Model search Started");
            StringBuffer sql = new StringBuffer("SELECT * FROM A_Book WHERE 1=1");

            if (bean != null) {
                    if (bean.getId() > 0) {
                            sql.append(" AND id = " + bean.getId());
                    }
                    if (bean.getFlightName() != null && bean.getFlightName().length()
> 0) {

                            sql.append(" AND FlightName like '" + bean.getFlightName()

+ "%'");
                    }
                    if (bean.getFirstName() != null && bean.getFirstName().length() >
0) {

                            sql.append(" AND FirstName like '" + bean.getFirstName() +

"%'");
                    }
                    if (bean.getLastName() != null && bean.getLastName().length() >
0) {

                            sql.append(" AND LastName like '" + bean.getLastName() +

"%'");
                    }
                    if (bean.getEmailId() != null && bean.getEmailId().length() > 0)
{

                            sql.append(" AND EmailId like '" + bean.getEmailId() +

"%'");
                    }

            }

            // if page size is greater than zero then apply pagination
            if (pageSize > 0) {
                    // Calculate start record index
                    pageNo = (pageNo - 1) * pageSize;

                    sql.append(" Limit " + pageNo + ", " + pageSize);
                    // sql.append(" limit " + pageNo + "," + pageSize);
            }
```

```
		System.out.println("user model search  :" + sql);
		ArrayList list = new ArrayList();
		Connection conn = null;
		try {
			conn = JDBCDataSource.getConnection();
			PreparedStatement pstmt = conn.prepareStatement(sql.toString());
			ResultSet rs = pstmt.executeQuery();
			while (rs.next()) {
				bean = new BookBean();
				bean.setId(rs.getLong(1));
				bean.setFlightId(rs.getLong(2));
				bean.setFlightName(rs.getString(3));
				bean.setFirstName(rs.getString(4));
				bean.setLastName(rs.getString(5));
				bean.setMobileNo(rs.getString(6));
				bean.setBookDate(rs.getDate(7));
				bean.setEmailId(rs.getString(8));
				bean.setAddress(rs.getString(9));
				bean.setNoOfPerson(rs.getLong(10));
				bean.setPrice(rs.getLong(11));
				bean.setFinalPrice(rs.getLong(12));
				bean.setCreatedBy(rs.getString(13));
				bean.setModifiedBy(rs.getString(14));
				bean.setCreatedDatetime(rs.getTimestamp(15));
				bean.setModifiedDatetime(rs.getTimestamp(16));

				list.add(bean);
			}
			rs.close();
		} catch (Exception e) {
			log.error("Database Exception..", e);
			throw new ApplicationException("Exception : Exception in search
user");
		} finally {
			JDBCDataSource.closeConnection(conn);
		}

		log.debug("Model search End");
		return list;
	}

	public List list() throws ApplicationException {
		return list(0, 0);
	}

	public List list(int pageNo, int pageSize) throws ApplicationException {
		log.debug("Model list Started");
		ArrayList list = new ArrayList();
		StringBuffer sql = new StringBuffer("select * from A_Book");
		// if page size is greater than zero then apply pagination
		if (pageSize > 0) {
			// Calculate start record index
			pageNo = (pageNo - 1) * pageSize;
			sql.append(" limit " + pageNo + "," + pageSize);
		}
		System.out.println("sql in list user :" + sql);
		Connection conn = null;

		try {
			conn = JDBCDataSource.getConnection();
			PreparedStatement pstmt = conn.prepareStatement(sql.toString());
			ResultSet rs = pstmt.executeQuery();
			while (rs.next()) {
				BookBean bean = new BookBean();
				bean.setId(rs.getLong(1));
				bean.setFlightId(rs.getLong(2));
				bean.setFlightName(rs.getString(3));
				bean.setFirstName(rs.getString(4));
				bean.setLastName(rs.getString(5));
				bean.setMobileNo(rs.getString(6));
				bean.setBookDate(rs.getDate(7));
				bean.setEmailId(rs.getString(8));
				bean.setAddress(rs.getString(9));
				bean.setNoOfPerson(rs.getLong(10));
				bean.setPrice(rs.getLong(11));
				bean.setFinalPrice(rs.getLong(12));
				bean.setCreatedBy(rs.getString(13));
				bean.setModifiedBy(rs.getString(14));
				bean.setCreatedDatetime(rs.getTimestamp(15));
				bean.setModifiedDatetime(rs.getTimestamp(16));
				list.add(bean);
			}
			rs.close();
		} catch (Exception e) {
			log.error("Database Exception..", e);
			throw new ApplicationException("Exception : Exception in getting 
list of users");
		} finally {
			JDBCDataSource.closeConnection(conn);
		}

		log.debug("Model list End");
		return list;

	}

}
```

FlightModel.java

```
package in.co.air.line.ticket.model;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;

import org.apache.log4j.Logger;

import in.co.air.line.ticket.bean.FlightBean;
import in.co.air.line.ticket.bean.UserBean;
import in.co.air.line.ticket.exception.ApplicationException;
import in.co.air.line.ticket.exception.DatabaseException;
import in.co.air.line.ticket.exception.DuplicateRecordException;
import in.co.air.line.ticket.util.JDBCDataSource;

public class FlightModel {

	private static Logger log = Logger.getLogger(FlightModel.class);

	public Integer nextPK() throws DatabaseException {
		log.debug("Model nextPK Started");
		Connection conn = null;
		int pk = 0;

		try {
			conn = JDBCDataSource.getConnection();
			PreparedStatement pstmt = conn.prepareStatement("SELECT MAX(ID)
FROM A_Flight");
			ResultSet rs = pstmt.executeQuery();
			while (rs.next()) {
				pk = rs.getInt(1);
			}
			rs.close();

		} catch (Exception e) {
			log.error("Database Exception..", e);
			throw new DatabaseException("Exception : Exception in getting
PK");
		} finally {
			JDBCDataSource.closeConnection(conn);
		}
		log.debug("Model nextPK End");
		return pk + 1;
	}

	public long add(FlightBean bean) throws ApplicationException,
DuplicateRecordException {

		Connection conn = null;
		int pk = 0;

		FlightBean existbean = findByName(bean.getFightName());

		if (existbean != null) {
			throw new DuplicateRecordException("Flight is already exists");
		}

		try {
			conn = JDBCDataSource.getConnection();
			pk = nextPK();
			// Get auto-generated next primary key
			System.out.println(pk + " in ModelJDBC");
			conn.setAutoCommit(false); // Begin transaction
			PreparedStatement pstmt = conn
					.prepareStatement("INSERT INTO A_Flight 
VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)");
			pstmt.setInt(1, pk);
			pstmt.setString(2, bean.getFlightNo());
			pstmt.setString(3, bean.getFightName());
			pstmt.setString(4, bean.getFromCity());
			pstmt.setString(5, bean.getToCity());
			pstmt.setDate(6, new java.sql.Date(bean.getDate().getTime()));
			pstmt.setString(7, bean.getDescription());
			pstmt.setString(8, bean.getTime());
			pstmt.setString(9, bean.getTravelDuraion());
			pstmt.setLong(10, bean.getTicketPrice());
			pstmt.setString(11, bean.getAirPortName());
			pstmt.setString(12, bean.getCreatedBy());
			pstmt.setString(13, bean.getModifiedBy());
			pstmt.setTimestamp(14, bean.getCreatedDatetime());
			pstmt.setTimestamp(15, bean.getModifiedDatetime());
			pstmt.executeUpdate();
			conn.commit(); // End transaction
			pstmt.close();
		} catch (Exception e) {

			try {
				conn.rollback();
			} catch (Exception ex) {
				ex.printStackTrace();
				throw new ApplicationException("Exception : add rollback 
exception " + ex.getMessage());
			}
			throw new ApplicationException("Exception : Exception in add 
User");
		} finally {
			JDBCDataSource.closeConnection(conn);
		}

		return pk;
	}

	public FlightBean findByName(String name) throws ApplicationException {
		log.debug("Model findByLogin Started");
		StringBuffer sql = new StringBuffer("SELECT * FROM A_Flight WHERE 
fightName=?");
		FlightBean bean = null;
		Connection conn = null;
		System.out.println("sql" + sql);

		try {
			conn = JDBCDataSource.getConnection();
			PreparedStatement pstmt = conn.prepareStatement(sql.toString());
			pstmt.setString(1, name);
			ResultSet rs = pstmt.executeQuery();
			while (rs.next()) {
				bean = new FlightBean();
				bean.setId(rs.getLong(1));
				bean.setFlightNo(rs.getString(2));
				bean.setFightName(rs.getString(3));
				bean.setFromCity(rs.getString(4));
				bean.setToCity(rs.getString(5));
				bean.setDate(rs.getDate(6));
				bean.setDescription(rs.getString(7));
				bean.setTime(rs.getString(8));
				bean.setTravelDuraion(rs.getString(9));
				bean.setTicketPrice(rs.getLong(10));
				bean.setAirPortName(rs.getString(11));
				bean.setCreatedBy(rs.getString(12));
				bean.setModifiedBy(rs.getString(13));
				bean.setCreatedDatetime(rs.getTimestamp(14));
				bean.setModifiedDatetime(rs.getTimestamp(15));

			}
			rs.close();
		} catch (Exception e) {
			e.printStackTrace();
			log.error("Database Exception..", e);
			throw new ApplicationException("Exception : Exception in getting 
User by login");
		} finally {
			JDBCDataSource.closeConnection(conn);
		}
		log.debug("Model findByLogin End");
		return bean;
	}

	public FlightBean findByPK(long pk) throws ApplicationException {
		log.debug("Model findByPK Started");
		StringBuffer sql = new StringBuffer("SELECT * FROM A_flight WHERE 
ID=?");
		FlightBean bean = null;
		Connection conn = null;

		try {
			conn = JDBCDataSource.getConnection();
			PreparedStatement pstmt = conn.prepareStatement(sql.toString());
			pstmt.setLong(1, pk);
			ResultSet rs = pstmt.executeQuery();
```

```
                    while (rs.next()) {
                        bean = new FlightBean();
                        bean.setId(rs.getLong(1));
                        bean.setFlightNo(rs.getString(2));
                        bean.setFightName(rs.getString(3));
                        bean.setFromCity(rs.getString(4));
                        bean.setToCity(rs.getString(5));
                        bean.setDate(rs.getDate(6));
                        bean.setDescription(rs.getString(7));
                        bean.setTime(rs.getString(8));
                        bean.setTravelDuraion(rs.getString(9));
                        bean.setTicketPrice(rs.getLong(10));
                        bean.setAirPortName(rs.getString(11));
                        bean.setCreatedBy(rs.getString(12));
                        bean.setModifiedBy(rs.getString(13));
                        bean.setCreatedDatetime(rs.getTimestamp(14));
                        bean.setModifiedDatetime(rs.getTimestamp(15));

                    }
                    rs.close();
                } catch (Exception e) {
                    e.printStackTrace();
                    log.error("Database Exception..", e);
                    throw new ApplicationException("Exception : Exception in getting
User by pk");
                } finally {
                    JDBCDataSource.closeConnection(conn);
                }
                log.debug("Model findByPK End");
                return bean;
        }

        public void delete(FlightBean bean) throws ApplicationException {

                Connection conn = null;
                try {
                    conn = JDBCDataSource.getConnection();
                    conn.setAutoCommit(false); // Begin transaction
                    PreparedStatement pstmt = conn.prepareStatement("DELETE FROM
A_flight WHERE ID=?");
                    pstmt.setLong(1, bean.getId());
                    pstmt.executeUpdate();
                    conn.commit(); // End transaction
                    pstmt.close();

                } catch (Exception e) {

                    try {
                        conn.rollback();
                    } catch (Exception ex) {
                        throw new ApplicationException("Exception : Delete
rollback exception " + ex.getMessage());
                    }
                    throw new ApplicationException("Exception : Exception in delete
User");
```

```
		} finally {
			JDBCDataSource.closeConnection(conn);
		}

	}

	public void update(FlightBean bean) throws ApplicationException,
DuplicateRecordException {
		log.debug("Model update Started");
		Connection conn = null;

		FlightBean beanExist = findByName(bean.getFightName());
		// Check if updated LoginId already exist
		if (beanExist != null && !(beanExist.getId() == bean.getId())) {
			throw new DuplicateRecordException("Flight is already exist");
		}

		try {
			conn = JDBCDataSource.getConnection();
			conn.setAutoCommit(false); // Begin transaction
			PreparedStatement pstmt = conn.prepareStatement(
					"UPDATE A_Flight SET
FlightNo=?,FightName=?,FromCity=?,ToCity=?,Date=?,Description=?,Time=?,TravelDuraion=
?,TicketPrice=?,AirPortName=?,"
								+
"CREATEDBY=?,MODIFIEDBY=?,CREATEDDATETIME=?,MODIFIEDDATETIME=? WHERE ID=?");
			pstmt.setString(1, bean.getFlightNo());
			pstmt.setString(2, bean.getFightName());
			pstmt.setString(3, bean.getFromCity());
			pstmt.setString(4, bean.getToCity());
			pstmt.setDate(5, new java.sql.Date(bean.getDate().getTime()));
			pstmt.setString(6, bean.getDescription());
			pstmt.setString(7, bean.getTime());
			pstmt.setString(8, bean.getTravelDuraion());
			pstmt.setLong(9, bean.getTicketPrice());
			pstmt.setString(10, bean.getAirPortName());
			pstmt.setString(11, bean.getCreatedBy());
			pstmt.setString(12, bean.getModifiedBy());
			pstmt.setTimestamp(13, bean.getCreatedDatetime());
			pstmt.setTimestamp(14, bean.getModifiedDatetime());
			pstmt.setLong(15, bean.getId());
			pstmt.executeUpdate();
			conn.commit(); // End transaction
			pstmt.close();
		} catch (Exception e) {
			e.printStackTrace();
			log.error("Database Exception..", e);
			try {
				conn.rollback();
			} catch (Exception ex) {
				throw new ApplicationException("Exception : Delete
rollback exception " + ex.getMessage());
			}
			throw new ApplicationException("Exception in updating User ");
		} finally {
```

```
            JDBCDataSource.closeConnection(conn);
        }
        log.debug("Model update End");
    }

    public List search(FlightBean bean) throws ApplicationException {
        return search(bean, 0, 0);
    }

    public List search(FlightBean bean, int pageNo, int pageSize) throws
ApplicationException {
        log.debug("Model search Started");
        StringBuffer sql = new StringBuffer("SELECT * FROM A_Flight WHERE 1=1");

        if (bean != null) {
            if (bean.getId() > 0) {
                sql.append(" AND id = " + bean.getId());
            }
            if (bean.getFightName() != null && bean.getFightName().length() >
0) {
                sql.append(" AND FightName like '" + bean.getFightName() +
"%'");
            }
            if (bean.getFlightNo() != null && bean.getFlightNo().length() >
0) {
                sql.append(" AND FlightNo like '" + bean.getFlightNo() +
"%'");
            }
            if (bean.getToCity() != null && bean.getToCity().length() > 0) {
                sql.append(" AND toCity like '" + bean.getToCity() +
"%'");
            }
            if (bean.getFromCity() != null && bean.getFromCity().length() >
0) {
                sql.append(" AND FromCity like '" + bean.getFromCity() +
"%'");
            }
            if (bean.getDate() != null && bean.getDate().~~getDate()~~ > 0) {
                sql.append(" AND Date = " + new
java.sql.Date(bean.getDate().getTime()));
            }

        }

        // if page size is greater than zero then apply pagination
        if (pageSize > 0) {
            // Calculate start record index
            pageNo = (pageNo - 1) * pageSize;

            sql.append(" Limit " + pageNo + ", " + pageSize);
            // sql.append(" limit " + pageNo + "," + pageSize);
        }

        System.out.println("user model search  :" + sql);
        ArrayList list = new ArrayList();
```

```
		Connection conn = null;
		try {
			conn = JDBCDataSource.getConnection();
			PreparedStatement pstmt = conn.prepareStatement(sql.toString());
			ResultSet rs = pstmt.executeQuery();
			while (rs.next()) {
				bean = new FlightBean();
				bean.setId(rs.getLong(1));
				bean.setFlightNo(rs.getString(2));
				bean.setFightName(rs.getString(3));
				bean.setFromCity(rs.getString(4));
				bean.setToCity(rs.getString(5));
				bean.setDate(rs.getDate(6));
				bean.setDescription(rs.getString(7));
				bean.setTime(rs.getString(8));
				bean.setTravelDuraion(rs.getString(9));
				bean.setTicketPrice(rs.getLong(10));
				bean.setAirPortName(rs.getString(11));
				bean.setCreatedBy(rs.getString(12));
				bean.setModifiedBy(rs.getString(13));
				bean.setCreatedDatetime(rs.getTimestamp(14));
				bean.setModifiedDatetime(rs.getTimestamp(15));

				list.add(bean);
			}
			rs.close();
		} catch (Exception e) {
			log.error("Database Exception..", e);
			throw new ApplicationException("Exception : Exception in search
user");
		} finally {
			JDBCDataSource.closeConnection(conn);
		}

		log.debug("Model search End");
		return list;
	}

	public List list() throws ApplicationException {
		return list(0, 0);
	}

	public List list(int pageNo, int pageSize) throws ApplicationException {
		log.debug("Model list Started");
		ArrayList list = new ArrayList();
		StringBuffer sql = new StringBuffer("select * from A_Flight");
		// if page size is greater than zero then apply pagination
		if (pageSize > 0) {
			// Calculate start record index
			pageNo = (pageNo - 1) * pageSize;
			sql.append(" limit " + pageNo + "," + pageSize);
		}

		System.out.println("sql in list user :" + sql);
		Connection conn = null;
		try {
			conn = JDBCDataSource.getConnection();
			PreparedStatement pstmt = conn.prepareStatement(sql.toString());
			ResultSet rs = pstmt.executeQuery();
			while (rs.next()) {
				FlightBean bean = new FlightBean();
				bean.setId(rs.getLong(1));
				bean.setFlightNo(rs.getString(2));
				bean.setFightName(rs.getString(3));
				bean.setFromCity(rs.getString(4));
				bean.setToCity(rs.getString(5));
				bean.setDate(rs.getDate(6));
				bean.setDescription(rs.getString(7));
				bean.setTime(rs.getString(8));
				bean.setTravelDuraion(rs.getString(9));
				bean.setTicketPrice(rs.getLong(10));
				bean.setAirPortName(rs.getString(11));
				bean.setCreatedBy(rs.getString(12));
				bean.setModifiedBy(rs.getString(13));
				bean.setCreatedDatetime(rs.getTimestamp(14));
				bean.setModifiedDatetime(rs.getTimestamp(15));

				list.add(bean);
			}
			rs.close();
		} catch (Exception e) {
			log.error("Database Exception..", e);
			throw new ApplicationException("Exception : Exception in getting
list of users");
		} finally {
			JDBCDataSource.closeConnection(conn);
		}

		log.debug("Model list End");
		return list;

	}

}
```

UserModel.java

```java
package in.co.air.line.ticket.model;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.Date;
import java.util.HashMap;
import java.util.List;
import org.apache.log4j.Logger;

import in.co.air.line.ticket.bean.UserBean;
import in.co.air.line.ticket.exception.ApplicationException;
import in.co.air.line.ticket.exception.DatabaseException;
import in.co.air.line.ticket.exception.DuplicateRecordException;
import in.co.air.line.ticket.exception.RecordNotFoundException;
import in.co.air.line.ticket.util.EmailBuilder;
import in.co.air.line.ticket.util.EmailMessage;
import in.co.air.line.ticket.util.EmailUtility;
import in.co.air.line.ticket.util.JDBCDataSource;


public class UserModel {
	private static Logger log = Logger.getLogger(UserModel.class);


	public Integer nextPK() throws DatabaseException {
		log.debug("Model nextPK Started");
		Connection conn = null;
		int pk = 0;

		try {
			conn = JDBCDataSource.getConnection();
			PreparedStatement pstmt = conn.prepareStatement("SELECT MAX(ID) 
FROM A_USER");
			ResultSet rs = pstmt.executeQuery();
			while (rs.next()) {
				pk = rs.getInt(1);
			}
			rs.close();

		} catch (Exception e) {
			log.error("Database Exception..", e);
			throw new DatabaseException("Exception : Exception in getting 
PK");
		} finally {
			JDBCDataSource.closeConnection(conn);
		}
		log.debug("Model nextPK End");
		return pk + 1;
	}

	public long add(UserBean bean) throws ApplicationException,
DuplicateRecordException {

		Connection conn = null;
		int pk = 0;

		UserBean existbean = findByLogin(bean.getLogin());

		if (existbean != null) {
			throw new DuplicateRecordException("Login Id already exists");
		}

		try {
			conn = JDBCDataSource.getConnection();
			pk = nextPK();
			// Get auto-generated next primary key
			System.out.println(pk + " in ModelJDBC");
			conn.setAutoCommit(false); // Begin transaction
			PreparedStatement pstmt = conn.prepareStatement("INSERT INTO
A_USER VALUES(?,?,?,?,?,?,?,?,?,?)");
			pstmt.setInt(1, pk);
			pstmt.setString(2, bean.getFirstName());
			pstmt.setString(3, bean.getLastName());
			pstmt.setString(4, bean.getLogin());
			pstmt.setString(5, bean.getPassword());
			pstmt.setLong(6, bean.getRoleId());
			pstmt.setString(7, bean.getCreatedBy());
			pstmt.setString(8, bean.getModifiedBy());
			pstmt.setTimestamp(9, bean.getCreatedDatetime());
			pstmt.setTimestamp(10, bean.getModifiedDatetime());
			pstmt.executeUpdate();
			conn.commit(); // End transaction
			pstmt.close();
		} catch (Exception e) {

			try {
				conn.rollback();
			} catch (Exception ex) {
				ex.printStackTrace();
				throw new ApplicationException("Exception : add rollback
exception " + ex.getMessage());
			}
			throw new ApplicationException("Exception : Exception in add
User");
		} finally {
			JDBCDataSource.closeConnection(conn);
		}

		return pk;
	}

	public void delete(UserBean bean) throws ApplicationException {
		Connection conn = null;
		try {
			conn = JDBCDataSource.getConnection();
			conn.setAutoCommit(false); // Begin transaction
			PreparedStatement pstmt = conn.prepareStatement("DELETE FROM 
A_USER WHERE ID=?");
			pstmt.setLong(1, bean.getId());
			pstmt.executeUpdate();
			conn.commit(); // End transaction
			pstmt.close();

		} catch (Exception e) {

			try {
				conn.rollback();
			} catch (Exception ex) {
				throw new ApplicationException("Exception : Delete 
rollback exception " + ex.getMessage());
			}
			throw new ApplicationException("Exception : Exception in delete 
User");
		} finally {
			JDBCDataSource.closeConnection(conn);
		}

	}

	public UserBean findByLogin(String login) throws ApplicationException {
		log.debug("Model findByLogin Started");
		StringBuffer sql = new StringBuffer("SELECT * FROM A_USER WHERE 
LOGIN=?");
		UserBean bean = null;
		Connection conn = null;
		System.out.println("sql" + sql);

		try {
			conn = JDBCDataSource.getConnection();
			PreparedStatement pstmt = conn.prepareStatement(sql.toString());
			pstmt.setString(1, login);
			ResultSet rs = pstmt.executeQuery();
			while (rs.next()) {
				bean = new UserBean();
				bean.setId(rs.getLong(1));
				bean.setFirstName(rs.getString(2));
				bean.setLastName(rs.getString(3));
				bean.setLogin(rs.getString(4));
				bean.setPassword(rs.getString(5));
				bean.setRoleId(rs.getLong(6));
				bean.setCreatedBy(rs.getString(7));
				bean.setModifiedBy(rs.getString(8));
				bean.setCreatedDatetime(rs.getTimestamp(9));
				bean.setModifiedDatetime(rs.getTimestamp(10));

			}
			rs.close();
```

```
            } catch (Exception e) {
                  e.printStackTrace();
                  log.error("Database Exception..", e);
                  throw new ApplicationException("Exception : Exception in getting
User by login");
            } finally {
                  JDBCDataSource.closeConnection(conn);
            }
            log.debug("Model findByLogin End");
            return bean;
      }


      public UserBean findByPK(long pk) throws ApplicationException {
            log.debug("Model findByPK Started");
            StringBuffer sql = new StringBuffer("SELECT * FROM A_USER WHERE ID=?");
            UserBean bean = null;
            Connection conn = null;

            try {
                  conn = JDBCDataSource.getConnection();
                  PreparedStatement pstmt = conn.prepareStatement(sql.toString());
                  pstmt.setLong(1, pk);
                  ResultSet rs = pstmt.executeQuery();
                  while (rs.next()) {
                        bean = new UserBean();
                        bean.setId(rs.getLong(1));
                        bean.setFirstName(rs.getString(2));
                        bean.setLastName(rs.getString(3));
                        bean.setLogin(rs.getString(4));
                        bean.setPassword(rs.getString(5));
                        bean.setRoleId(rs.getLong(6));
                        bean.setCreatedBy(rs.getString(7));
                        bean.setModifiedBy(rs.getString(8));
                        bean.setCreatedDatetime(rs.getTimestamp(9));
                        bean.setModifiedDatetime(rs.getTimestamp(10));

                  }
                  rs.close();
            } catch (Exception e) {
                  e.printStackTrace();
                  log.error("Database Exception..", e);
                  throw new ApplicationException("Exception : Exception in getting
User by pk");
            } finally {
                  JDBCDataSource.closeConnection(conn);
            }
            log.debug("Model findByPK End");
            return bean;
      }

      public void update(UserBean bean) throws ApplicationException,
DuplicateRecordException {
            log.debug("Model update Started");
            Connection conn = null;
```

```
			UserBean beanExist = findByLogin(bean.getLogin());
			// Check if updated LoginId already exist
			if (beanExist != null && !(beanExist.getId() == bean.getId())) {
				throw new DuplicateRecordException("LoginId is already exist");
			}

			try {
				conn = JDBCDataSource.getConnection();
				conn.setAutoCommit(false); // Begin transaction
				PreparedStatement pstmt = conn.prepareStatement(
						"UPDATE A_USER SET 
FIRSTNAME=?,LASTNAME=?,LOGIN=?,PASSWORD=?,ROLEID=?,"
						+ 
"CREATEDBY=?,MODIFIEDBY=?,CREATEDDATETIME=?,MODIFIEDDATETIME=? WHERE ID=?");
				pstmt.setString(1, bean.getFirstName());
				pstmt.setString(2, bean.getLastName());
				pstmt.setString(3, bean.getLogin());
				pstmt.setString(4, bean.getPassword());
				pstmt.setLong(5, bean.getRoleId());
				pstmt.setString(6, bean.getCreatedBy());
				pstmt.setString(7, bean.getModifiedBy());
				pstmt.setTimestamp(8, bean.getCreatedDatetime());
				pstmt.setTimestamp(9, bean.getModifiedDatetime());
				pstmt.setLong(10, bean.getId());
				pstmt.executeUpdate();
				conn.commit(); // End transaction
				pstmt.close();
			} catch (Exception e) {
				e.printStackTrace();
				log.error("Database Exception..", e);
				try {
					conn.rollback();
				} catch (Exception ex) {
					throw new ApplicationException("Exception : Delete 
rollback exception " + ex.getMessage());
				}
				throw new ApplicationException("Exception in updating User ");
			} finally {
				JDBCDataSource.closeConnection(conn);
			}
			log.debug("Model update End");
		}


		public List search(UserBean bean) throws ApplicationException {
			return search(bean, 0, 0);
		}

		public List search(UserBean bean, int pageNo, int pageSize) throws 
ApplicationException {
			log.debug("Model search Started");
			StringBuffer sql = new StringBuffer("SELECT * FROM A_USER WHERE 1=1");

			if (bean != null) {
```

```
                    if (bean.getId() > 0) {
                          sql.append(" AND id = " + bean.getId());
                    }
                    if (bean.getFirstName() != null && bean.getFirstName().length() >
0) {

                          sql.append(" AND FIRSTNAME like '" + bean.getFirstName() +

"%'");
                    }
                    if (bean.getLastName() != null && bean.getLastName().length() >
0) {

                          sql.append(" AND LASTNAME like '" + bean.getLastName() +

"%'");
                    }
                    if (bean.getLogin() != null && bean.getLogin().length() > 0) {
                          sql.append(" AND LOGIN like '" + bean.getLogin() + "%'");
                    }
                    if (bean.getPassword() != null && bean.getPassword().length() >
0) {

                          sql.append(" AND PASSWORD like '" + bean.getPassword() +

"%'");
                    }

                    if (bean.getRoleId() > 0) {
                          sql.append(" AND ROLEID = " + bean.getRoleId());
                    }



             }

             // if page size is greater than zero then apply pagination
             if (pageSize > 0) {
                    // Calculate start record index
                    pageNo = (pageNo - 1) * pageSize;

                    sql.append(" Limit " + pageNo + ", " + pageSize);
                    // sql.append(" limit " + pageNo + "," + pageSize);
             }

             System.out.println("user model search  :"+sql);
             ArrayList list = new ArrayList();
             Connection conn = null;
             try {
                    conn = JDBCDataSource.getConnection();
                    PreparedStatement pstmt = conn.prepareStatement(sql.toString());
                    ResultSet rs = pstmt.executeQuery();
                    while (rs.next()) {
                          bean = new UserBean();
                          bean.setId(rs.getLong(1));
                          bean.setFirstName(rs.getString(2));
                          bean.setLastName(rs.getString(3));
                          bean.setLogin(rs.getString(4));
                          bean.setPassword(rs.getString(5));
                          bean.setRoleId(rs.getLong(6));
                          bean.setCreatedBy(rs.getString(7));
```

```
                                bean.setModifiedBy(rs.getString(8));
                                bean.setCreatedDatetime(rs.getTimestamp(9));
                                bean.setModifiedDatetime(rs.getTimestamp(10));

                                list.add(bean);
                        }
                        rs.close();
                } catch (Exception e) {
                        log.error("Database Exception..", e);
                        throw new ApplicationException("Exception : Exception in search
user");
                } finally {
                        JDBCDataSource.closeConnection(conn);
                }

                log.debug("Model search End");
                return list;
        }

        public List list() throws ApplicationException {
                return list(0, 0);
        }

        public List list(int pageNo, int pageSize) throws ApplicationException {
                log.debug("Model list Started");
                ArrayList list = new ArrayList();
                StringBuffer sql = new StringBuffer("select * from A_USER");
                // if page size is greater than zero then apply pagination
                if (pageSize > 0) {
                        // Calculate start record index
                        pageNo = (pageNo - 1) * pageSize;
                        sql.append(" limit " + pageNo + "," + pageSize);
                }


                System.out.println("sql in list user :"+sql);
                Connection conn = null;

                try {
                        conn = JDBCDataSource.getConnection();
                        PreparedStatement pstmt = conn.prepareStatement(sql.toString());
                        ResultSet rs = pstmt.executeQuery();
                        while (rs.next()) {
                                UserBean bean = new UserBean();
                                bean.setId(rs.getLong(1));
                                bean.setFirstName(rs.getString(2));
                                bean.setLastName(rs.getString(3));
                                bean.setLogin(rs.getString(4));
                                bean.setPassword(rs.getString(5));
                                bean.setRoleId(rs.getLong(6));
                                bean.setCreatedBy(rs.getString(7));
                                bean.setModifiedBy(rs.getString(8));
                                bean.setCreatedDatetime(rs.getTimestamp(9));
                                bean.setModifiedDatetime(rs.getTimestamp(10));
```

```
                    list.add(bean);
                }
                rs.close();
            } catch (Exception e) {
                log.error("Database Exception..", e);
                throw new ApplicationException("Exception : Exception in getting
list of users");
            } finally {
                JDBCDataSource.closeConnection(conn);
            }

            log.debug("Model list End");
            return list;

        }

        public UserBean authenticate(String login, String password) throws
ApplicationException {
            log.debug("Model authenticate Started");
            StringBuffer sql = new StringBuffer("SELECT * FROM A_USER WHERE LOGIN =
? AND PASSWORD = ?");
            UserBean bean = null;
            Connection conn = null;

            try {
                conn = JDBCDataSource.getConnection();
                PreparedStatement pstmt = conn.prepareStatement(sql.toString());
                pstmt.setString(1, login);
                pstmt.setString(2, password);
                ResultSet rs = pstmt.executeQuery();
                while (rs.next()) {
                    bean = new UserBean();
                    bean.setId(rs.getLong(1));
                    bean.setFirstName(rs.getString(2));
                    bean.setLastName(rs.getString(3));
                    bean.setLogin(rs.getString(4));
                    bean.setPassword(rs.getString(5));
                    bean.setRoleId(rs.getLong(6));
                    bean.setCreatedBy(rs.getString(7));
                    bean.setModifiedBy(rs.getString(8));
                    bean.setCreatedDatetime(rs.getTimestamp(9));
                    bean.setModifiedDatetime(rs.getTimestamp(10));
                    System.out.println("Usermodel here");
                }
            } catch (Exception e) {
                log.error("Database Exception..", e);
                throw new ApplicationException("Exception : Exception in get
roles");

            } finally {
                JDBCDataSource.closeConnection(conn);
            }

            log.debug("Model authenticate End");
            return bean;
```

```
	}

	public List getRoles(UserBean bean) throws ApplicationException {
		log.debug("Model get roles Started");
		StringBuffer sql = new StringBuffer("SELECT * FROM A_USER WHERE 
roleId=?");
		Connection conn = null;
		List list = new ArrayList();
		try {

			conn = JDBCDataSource.getConnection();
			PreparedStatement pstmt = conn.prepareStatement(sql.toString());
			pstmt.setLong(1, bean.getRoleId());
			ResultSet rs = pstmt.executeQuery();
			while (rs.next()) {
				bean = new UserBean();
				bean.setId(rs.getLong(1));
				bean.setFirstName(rs.getString(2));
				bean.setLastName(rs.getString(3));
				bean.setLogin(rs.getString(4));
				bean.setPassword(rs.getString(5));
				bean.setRoleId(rs.getLong(6));
				bean.setCreatedBy(rs.getString(7));
				bean.setModifiedBy(rs.getString(8));
				bean.setCreatedDatetime(rs.getTimestamp(9));
				bean.setModifiedDatetime(rs.getTimestamp(10));

				list.add(bean);
			}
			rs.close();
		} catch (Exception e) {
			log.error("Database Exception..", e);
			throw new ApplicationException("Exception : Exception in get 
roles");

		} finally {
			JDBCDataSource.closeConnection(conn);
		}
		log.debug("Model get roles End");
		return list;
	}


		public boolean changePassword(Long id, String oldPassword, String 
newPassword)
				throws RecordNotFoundException, ApplicationException {

			log.debug("model changePassword Started");

			boolean flag = false;

			UserBean beanExist = null;

			beanExist = findByPK(id);
			if (beanExist != null &&
beanExist.getPassword().equals(oldPassword)) {
				beanExist.setPassword(newPassword);
				try {
					update(beanExist);
				} catch (DuplicateRecordException e) {
					log.error(e);
					throw new ApplicationException("LoginId is already
exist");
				}
				flag = true;
			} else {
				throw new RecordNotFoundException("Old password is
Invalid");
			}

			HashMap<String, String> map = new HashMap<String, String>();

			map.put("login", beanExist.getLogin());
			map.put("password", beanExist.getPassword());
			map.put("firstName", beanExist.getFirstName());
			map.put("lastName", beanExist.getLastName());

			String message = EmailBuilder.getChangePasswordMessage(map);

			EmailMessage msg = new EmailMessage();

			msg.setTo(beanExist.getLogin());
			msg.setSubject("SUNARYS ORS Password has been changed
Successfully.");
			msg.setMessage(message);
			msg.setMessageType(EmailMessage.HTML_MSG);

			try {
				EmailUtility.sendMail(msg);
			} catch (Exception e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			}

			log.debug("Model changePassword End");
			return flag;

		}

	public long registerUser(UserBean bean)
			throws ApplicationException, DuplicateRecordException {

		log.debug("Model add Started");

		long pk = add(bean);

		HashMap<String, String> map = new HashMap<String, String>();
		map.put("login", bean.getLogin());
		map.put("password", bean.getPassword());
		String message = EmailBuilder.getUserRegistrationMessage(map);

		EmailMessage msg = new EmailMessage();

		msg.setTo(bean.getLogin());
		msg.setSubject("Registration is successful for ORS Project SunilOS");
		msg.setMessage(message);
		msg.setMessageType(EmailMessage.HTML_MSG);

		try {
			EmailUtility.sendMail(msg);
		} catch (Exception e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
		return pk;
	}

	public boolean forgetPassword(String login)
			throws ApplicationException, RecordNotFoundException,
ApplicationException {
		UserBean userData = findByLogin(login);

		boolean flag = false;

		if (userData == null) {
			throw new RecordNotFoundException("Email ID does not exists !");

		}

		HashMap<String, String> map = new HashMap<String, String>();
		map.put("login", userData.getLogin());
		map.put("password", userData.getPassword());
		map.put("firstName", userData.getFirstName());
		map.put("lastName", userData.getLastName());
		String message = EmailBuilder.getForgetPasswordMessage(map);
		EmailMessage msg = new EmailMessage();
		msg.setTo(login);
		msg.setSubject("SUNARYS ORS Password reset");
		msg.setMessage(message);
		msg.setMessageType(EmailMessage.HTML_MSG);
		EmailUtility.sendMail(msg);
		flag = true;

		return flag;
	}
}
```

ApplicationException.java

```
package in.co.air.line.ticket.exception;



public class ApplicationException  extends Exception
{

	public ApplicationException(String msg) {
		super(msg);
	}
}
```

DataException.java

```
package in.co.air.line.ticket.exception;



public class DatabaseException  extends Exception
{

   public DatabaseException(String msg) {
       super(msg);
   }
}
```

Conclusion

Further enhancements to the application can be made which may include:

- The further improvement may include the cross Database connectivity in order to provide any person to book ticket from any Flight host.