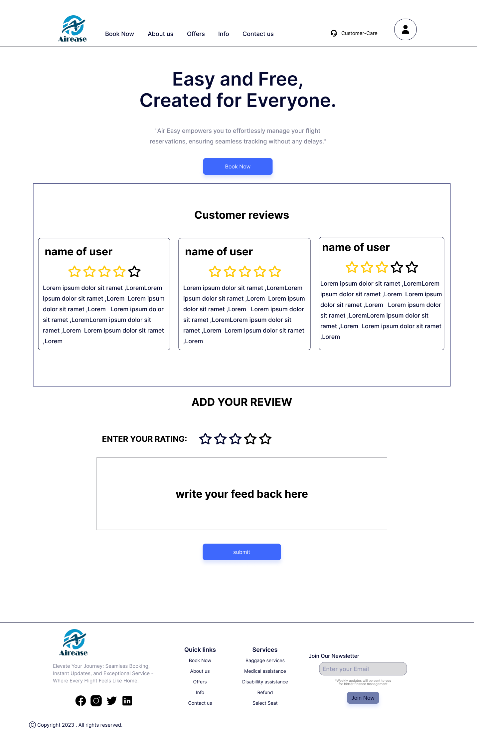
A screenshot of a website

Description automatically generated**HOME PAGE:**

No Apis needed for this page.

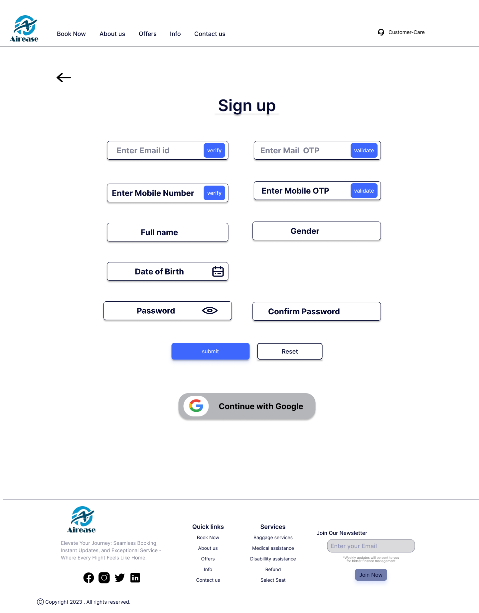


**ABOUT US PAGE:**

(1) POST API to add feedback

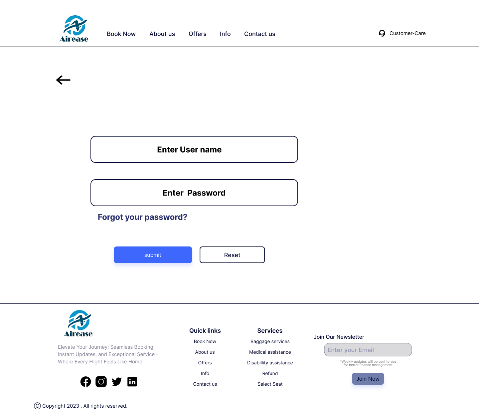
(2) GET API to show previous feedback

of other customers.



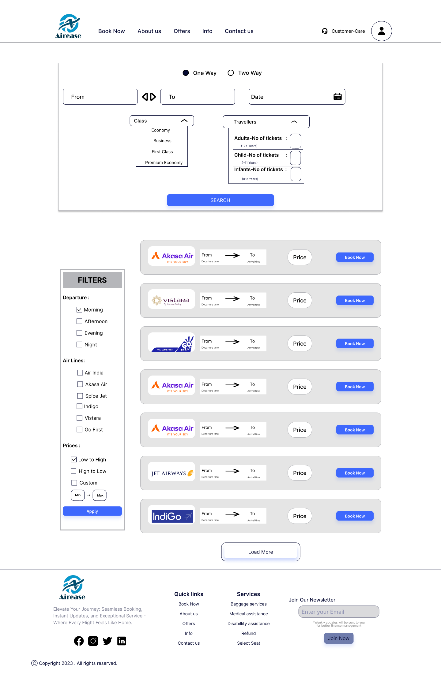
**SIGNUP PAGE:**

1. POST API to add customer.



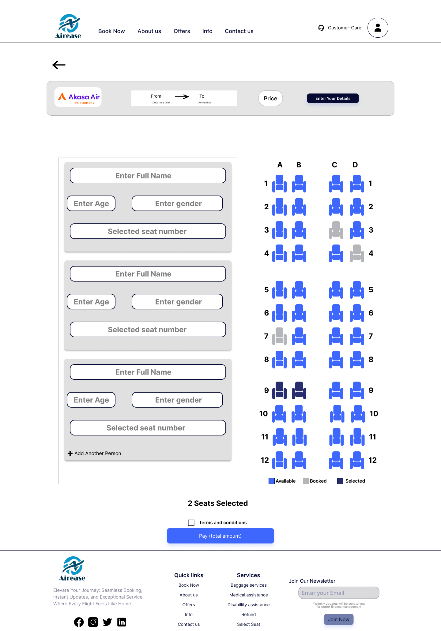
**USER LOGIN:**

1. GET API TO LOGIN



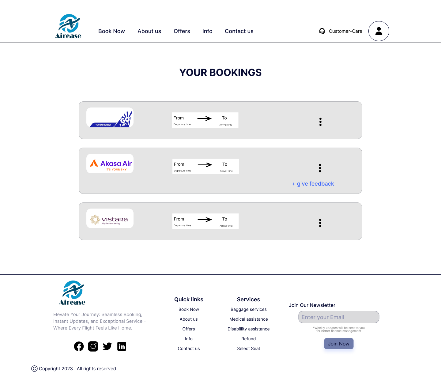
**SEARCH FLIGHTS PAGE:**

1. GET API to get flights list.



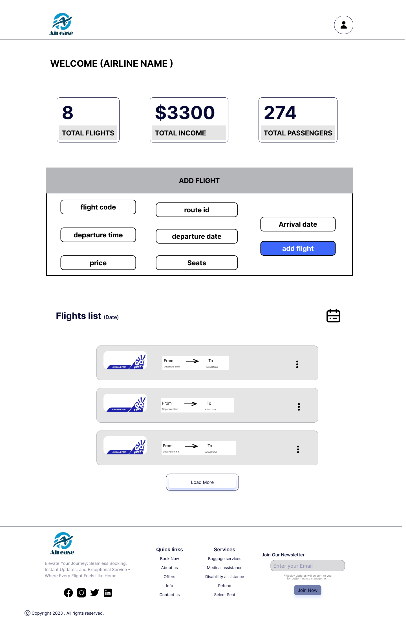
**BOOKING PAGE:**

1. POST API to book tickets



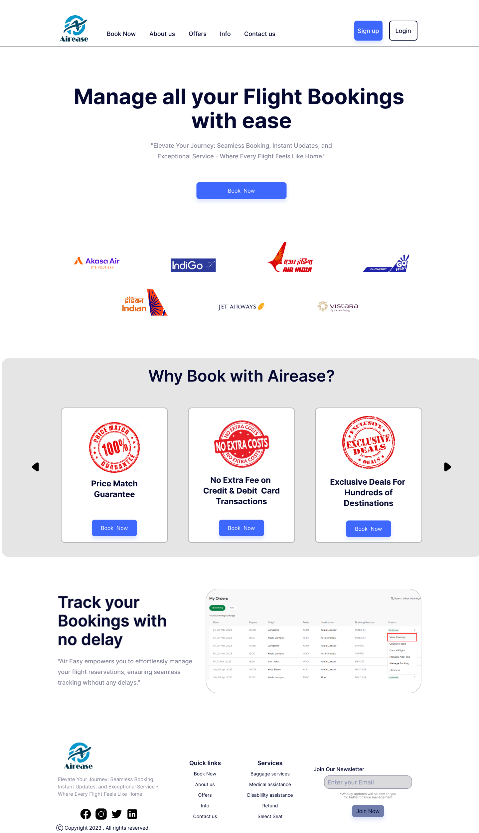
**PREVIOUS BOOKING:**

1. GET API to get bookings.
2. DELETE API to cancel booking.



**AIRLINE DASHBOARD:**

1. GET API to get total no of flights, income, passengers.
2. GET API to get list of flights from that airline.
3. PUT API to update flight details.
4. DELETE API to remove flight.
5. POST API to add new flights.



A screenshot of a review

Description automatically generated

**POST API to add feedback:**

@PostMapping("/feedback/{cid}/{fid}")

**public** ResponseEntity<?> WriteFeedback(@PathVariable("cid") **int** cid, @PathVariable("fid") **int** fid,

@RequestBody Feedback feedback) {

**try** {

Customer customer = customerService.getCustomer(cid);

Flight flight = flightService.getById(fid).

feedback. setCustomer(customer).

feedback. setFlight(flight).

feedback.setDate(LocalDate.*now*());

feedbackService.insert(feedback).

**return** ResponseEntity.*ok*().body(feedback);

} **catch** (InvalidIDException e) {

**return** ResponseEntity.*badRequest*().body(e.getMessage());

}

**GET API to show previous feedback of other customers:**

@GetMapping("/feedback/getall")

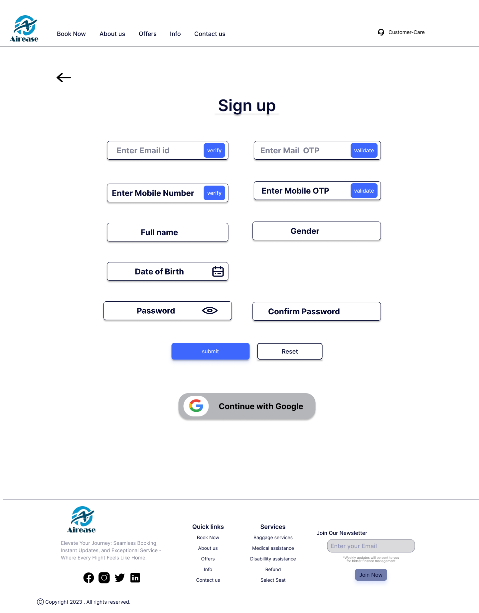
**public** List<Feedback> getAllFeedback (

@RequestParam (value = "page", required = **false**, defaultValue = "0") Integer page,

@RequestParam (value = "size", required = **false**, defaultValue = "10000000") Integer size) {

Pageable pageable = PageRequest.*of*(page, size)

**return** feedbackService.getAll(pageable);



**POST API to add customer.**

@PostMapping("/customer/signup")

**public** Customer signUp (@RequestBody Customer customer) {

User user = customer.getUser();

String passwordPlain = user.getPassword();

String encodedPassword = passwordEncoder.encode(passwordPlain);

user.setPassword(encodedPassword);

user.setRole("CUSTOMER");

user = userService.insert(user);

customer.setUser(user);

**return** customerService.insert(customer);}

A screenshot of a login page

Description automatically generated

**GET API TO LOGIN**

@GetMapping("/user/login")

**public** User login (Principal principal) {

String username = principal. getName ();

User user = (User)userService.loadUserByUsername(username);

**return** user;}

A screenshot of a computer

Description automatically generated

**GET API to get flights list.**

@GetMapping("/flight/all") // get all flights

**public** List<Flight> getAllFlights(@RequestParam(value = "page", required = **false**, defaultValue = "0") Integer page,

@RequestParam(value = "size", required = **false**, defaultValue = "1000000") Integer size) {

Pageable pageable = PageRequest.*of*(page, size);

**return** flightService.getAllFlights(pageable);

}

**GET API to get flights list based on source, destination, date:**

@GetMapping("/flight/{source}/{destination}/{date}")

// get flights by source and destination

**public** List<Flight> getFlightsbysdd(@PathVariable("source") String source,@PathVariable("destination") String destination,

@PathVariable("date”) @DateTimeFormat (iso = DateTimeFormat.ISO.***DATE***) LocalDate date) {

List<Flight> list = flightService.findby(source, destination, date);

**return** list;}

**GET API to get flights list based on filter:**

@GetMapping("/get/filter ? airline =")

**public** List<Flight> getflightsWithFilters (@RequestParam(required = **false**) String airline) {

**return** flightRepository.findAll().stream()

.filter(flight -> airline == **null** || (flight.getAirline() != **null** && flight.getAirline().getName().equals(airline)))

.collect(Collectors.*toList*());

}

A screenshot of a computer

Description automatically generated

**POST API to book tickets**

@PostMapping("/book/{cid}/{fid}")

**public** ResponseEntity<?> booktickets(@PathVariable("fid") **int** fid, @PathVariable("cid") **int** cid,

@RequestBody List<PassengerDto> passengerDtoList) {

**try** {

Customer customer = customerService.getCustomer(cid);

Flight flight = flightService.getById(fid);

List<CustomerFlight> bookedTickets = **new** ArrayList<>();

**double** totalPrice = 0;

**for** (PassengerDto passengerDto : passengerDtoList) {

CustomerFlight customerFlight = **new** CustomerFlight();

customerFlight.setCustomer(customer);

customerFlight.setFlight(flight);

customerFlight.setDate(LocalDate.*now*());

// Set passenger-specific information from the DTO

customerFlight.setName(passengerDto.getName());

customerFlight.setAge(passengerDto.getAge());

customerFlight.setGender(passengerDto.getGender());

customerFlight.setSeatclass(passengerDto.getSeatclass()); customerFlight.setPrice

(customerFlightService.price(fid,passengerDto.getAge(),passengerDto.getSeatclass()));

customerFlight.setSeatNumber(passengerDto.getSeatNumber());

totalPrice = totalPrice + (customerFlight.getPrice());

// Add the processed ticket to the list

bookedTickets.add(customerFlightService.insert(customerFlight));

}

Map<String, Object> response = **new** HashMap<>();

response.put("bookedTickets", bookedTickets);

response.put("totalPrice", totalPrice);

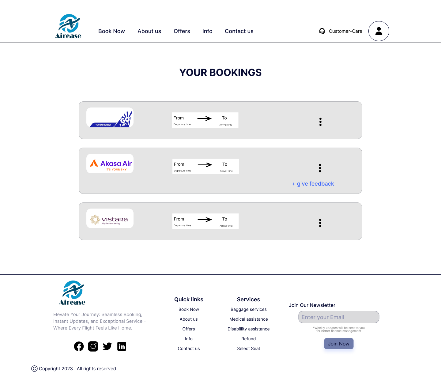
**return** ResponseEntity.*ok*().body(response);

} **catch** (InvalidIDException e) {

**return** ResponseEntity.*badRequest*().body(e.getMessage());

}

}



**GET API to get bookings.**

@GetMapping("/bookings/{cid}")

**public** List<Customer> getAllCustomer(

@RequestParam(value = "page", required = **false**, defaultValue = "0") Integer page,

@RequestParam (value = "size", required = **false**, defaultValue = "10000000") Integer size) {

Pageable pageable = PageRequest.*of*(page, size);

**return** customerService.getAll(pageable);}

**DELETE API to cancel booking.**

@DeleteMapping("/booking/delete/{bid}")

**public** ResponseEntity<?> deleteBooking(@PathVariable("bid") **int** id) {

**try** {

CustomerFlight customerFlight = customerFlightService.getBooking(id);

customerFlightService.deleteCustomer(customerFlight);

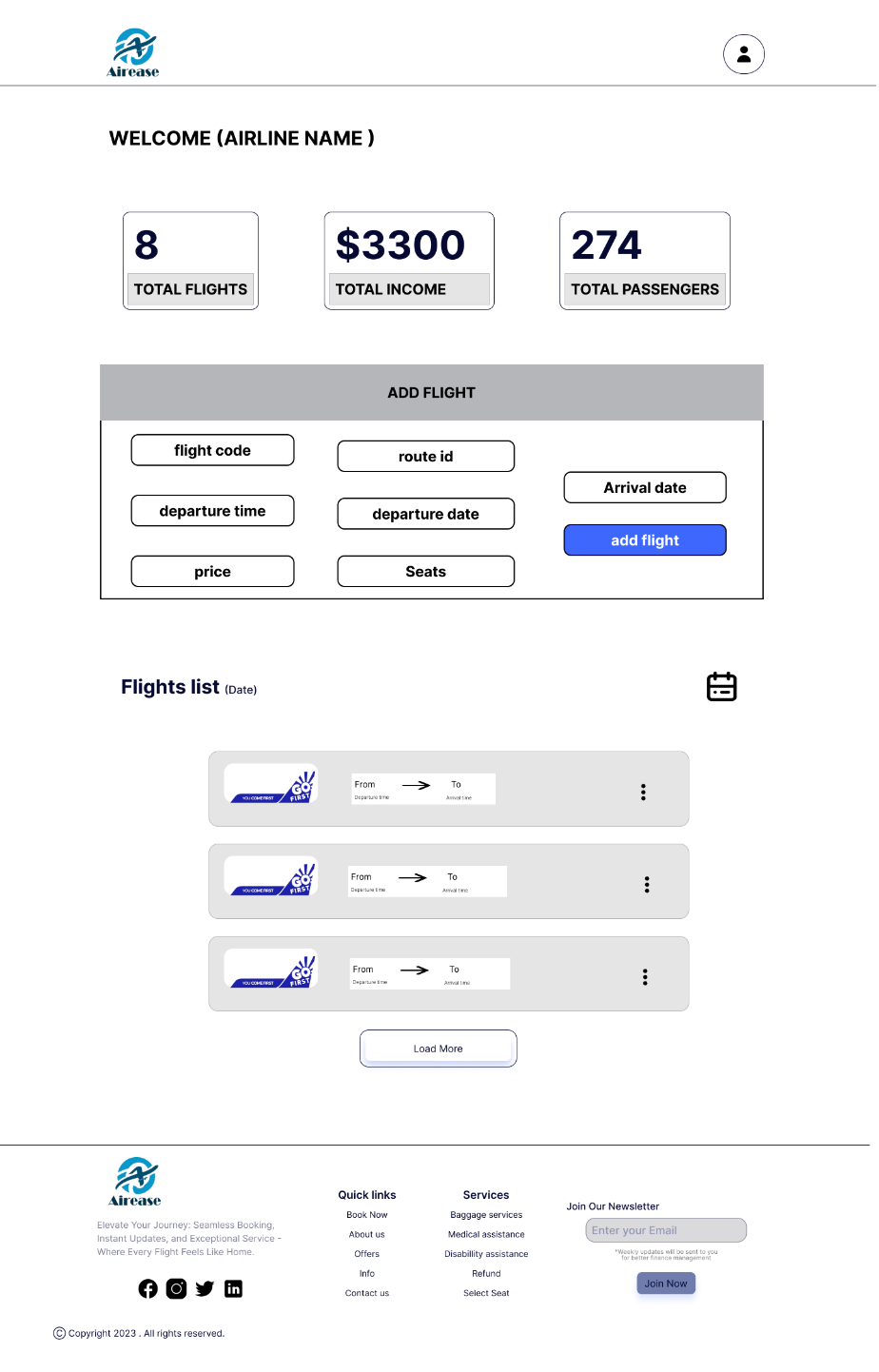
**return**

ResponseEntity.*ok*(). body("Booking deleted successfully");

} **catch** (InvalidIDException e) {

**return** ResponseEntity.*badRequest*().body(e.getMessage());

} }



**GET API to get list of flights from that airline.**

@GetMapping("/flight/getbyairline/{aid}") // get flights by airline id

**public** ResponseEntity<?> getFlight(@PathVariable("aid") **int** aid) {

**try** {

Airline airline = airlineService.getAirline(aid);

List<Flight> list = flightService.getFlightsByAirline(aid);

**return** ResponseEntity.*ok*(). body(list);

} **catch** (InvalidIDException e) {

**return** ResponseEntity.*badRequest*().body(e.getMessage());

}

}

**PUT API to update flight details.**

@PutMapping("/update/{id}") // update flight

**public** ResponseEntity<?> updateFlight(@PathVariable("id") **int** id, @RequestBody Flight flight) {

**try** {

Flight flight1 = flightService.getById(id);

**if** (flight.getCode() != **null**)

flight1.setCode(flight.getCode());

**if** (flight.getDepartureTime() != **null**)

flight1.setDepartureTime(flight.getDepartureTime());

**if** (flight.getDepartureDate() != **null**)

flight1.setDepartureDate(flight.getDepartureDate());

**if** (flight.getArrivalDate() != **null**)

flight1.setArrivalDate(flight.getArrivalDate());

**if** (flight.getAvailableSeats() != 0)

flight1.setAvailableSeats(flight.getAvailableSeats());

**if** (flight.getBusinessClassPrice() != 0)

flight1.setBusinessClassPrice(flight.getBusinessClassPrice());

**if** (flight.getEconomyClassPrice() != 0)

flight1.setEconomyClassPrice(flight.getEconomyClassPrice());

**if** (flight.getFirstClassPrice() != 0)

flight1.setFirstClassPrice(flight.getFirstClassPrice());

// Save the updated flight to the service

flight1 = flightService.insert(flight1);

**return** ResponseEntity.*ok*().body(flight1);

} **catch** (InvalidIDException e) {

**return** ResponseEntity.*badRequest*().body(e.getMessage());

}

}

**DELETE API to remove flight.**

@DeleteMapping("/flight/{fid}")//delete flight by flight id

**public** ResponseEntity<?> deleteflight(@PathVariable("fid") **int** fid) {

**try** {

Flight flight = flightService.getById(fid);

flightService.deleteFlight(fid);

**return** ResponseEntity.*ok*().body("flight deleted successfully");

} **catch** (InvalidIDException e) {

**return** ResponseEntity.*badRequest*().body(e.getMessage());

}

}

**POST API to add new flights.**

@PostMapping("/add/{aid}")

**public** ResponseEntity<?> addFlight(@PathVariable("aid") **int** aid, @RequestBody FlightDto flightDto) {

**try** {

Flight flight = **new** Flight();

Airline airline = airlineService.getAirline(aid);

flight.setAirline(airline);

String source = flightDto.getSource();

String destination = flightDto.getDestination();

Route route = routeService.getidbySD(source, destination);

flight.setRoute(route);

flight.setCode(flightDto.getCode());

flight.setDepartureTime(flightDto.getDepartureTime());

flight.setDepartureDate(flightDto.getDepartureDate());

flight.setArrivalDate(flightDto.getArrivalDate());

flight.setEconomyClassPrice(flightDto.getEconomyClassPrice());

flight.setFirstClassPrice(flightDto.getFirstClassPrice());

flight.setBusinessClassPrice(flightDto.getBusinessClassPrice());

flight.setAvailableSeats(flightDto.getAvailableSeats());

flight = flightService.insert(flight);

**return** ResponseEntity.*ok*().body(flight);

} **catch** (InvalidIDException e) {

**return** ResponseEntity.*badRequest*().body(e.getMessage());

}

}

**GET API to get passenger list of each flight.**

@GetMapping("/flight/passengers/{fid}") // get your bookings

**public** ResponseEntity<?> getpassengers(@PathVariable("fid") **int** fid) {

**try** {

Flight flight = flightService.getById(fid);

List<CustomerFlight> list = customerFlightService.getpassenger(fid);

**return** ResponseEntity.*ok*().body(list);

} **catch** (InvalidIDException e) {

**return** ResponseEntity.*badRequest*().body(e.getMessage());

}

}

**GET API to get total no of flights, income, passengers.**

@GetMapping("/statistics/flights/{aid}")

**public** ResponseEntity<?> getTotalFlights(@PathVariable("aid")**int** aid) {

**int** totalFlights = customerFlightService.getTotalFlights(aid);

**return** ResponseEntity.*ok*(totalFlights);

}

@GetMapping("/statistics/income/{aid}")

**public** ResponseEntity<Double> getTotalIncome(@PathVariable("aid")**int** airlineId) {

Double totalIncome = customerFlightService.getTotalIncome(airlineId);

**return** ResponseEntity.*ok*(totalIncome != **null** ? totalIncome : 0);

}

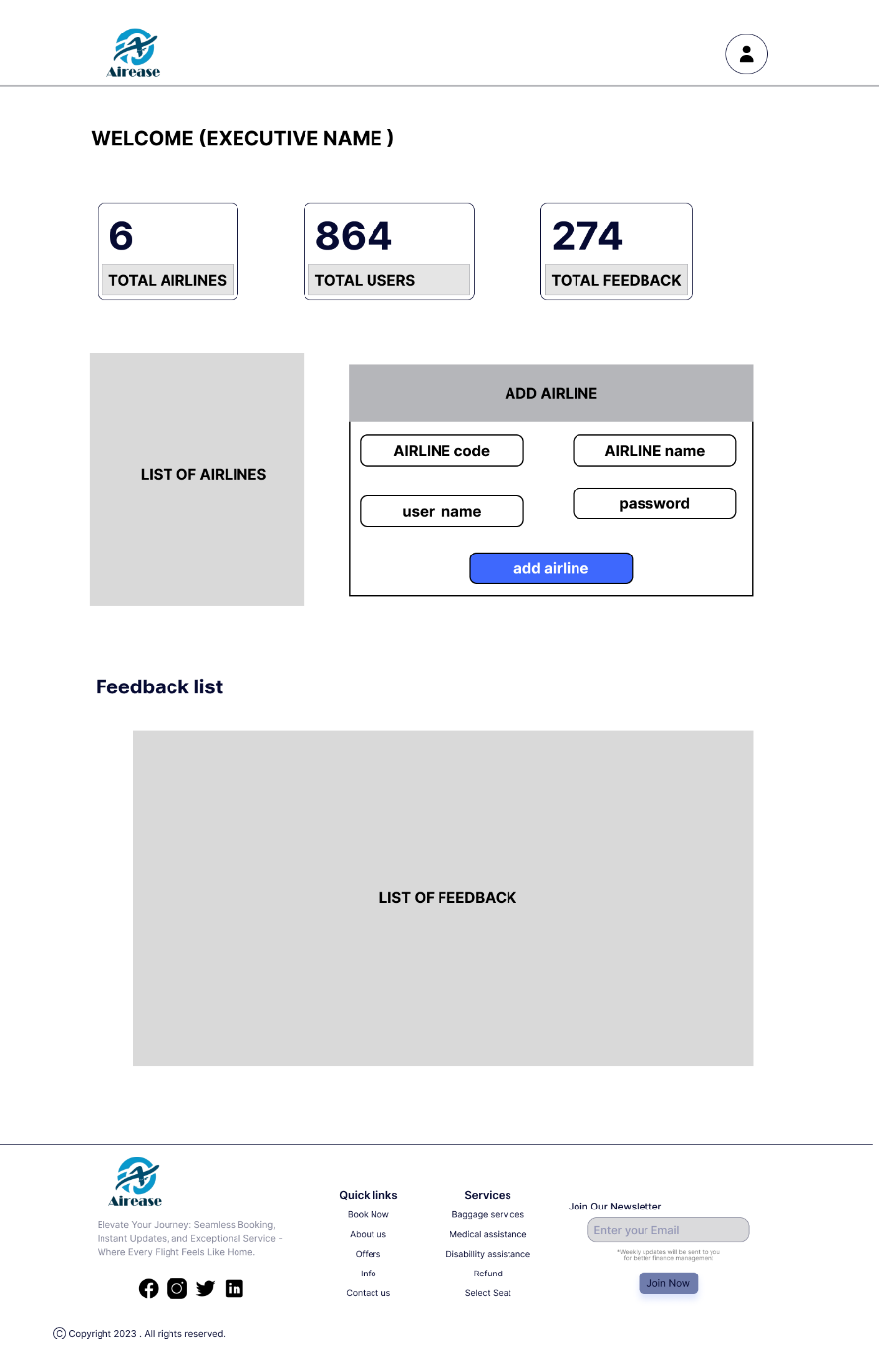
@GetMapping("/statistics/passengers/{aid}")

**public** ResponseEntity<Long> getTotalPassengers(@PathVariable("aid")**int** aid ) {

**long** totalPassengers = customerFlightService.getTotalPassengers( aid);

**return** ResponseEntity.*ok*(totalPassengers);

}

****

**GET API to see list of airlines.**

@GetMapping("/airline/getall")

**public** List<Airline> getAllAirline(@RequestParam(value = "page", required = **false**, defaultValue = "0") Integer page,

@RequestParam (value = "size", required = **false**, defaultValue = "10000000") Integer size) {

Pageable pageable = PageRequest.*of*(page, size); // null null

**return** airlineService.getAll(pageable);

}

**POST API to add new airlines.**

@PostMapping("/airline/add")

**public** Airline addAirline(@RequestBody Airline airline) {

User user = airline.getUser();

String passwordPlain = user.getPassword();

String encodedPassword = passwordEncoder.encode(passwordPlain);

user.setPassword(encodedPassword);

user.setRole("AIRLINE");

user = userService.insert(user);

airline.setUser(user);

**return** airlineService.insert(airline); }

**GET API to get list of FEEDBACK.**

@GetMapping("/feedback/getall")

**public** List<Feedback> getAllFeedback(

@RequestParam(value = "page", required = **false**, defaultValue = "0") Integer page,

@RequestParam(value = "size", required = **false**, defaultValue = "10000000") Integer size) {

Pageable pageable = PageRequest.*of*(page, size); // null null

//Pageable pageable = PageRequest.of(page, size, sort)

**return** feedbackService.getAll(pageable);

}

**GET API to get total no of airline, users, feedbacks.**

@GetMapping("/totalairlines")

**public** ResponseEntity<Long> getTotalAirlines() {

**long** totalAirlines = executiveService.getTotalAirlines();

**return** ResponseEntity.*ok*(totalAirlines);

}

//localhost:8081/executive/totalusers

@GetMapping("/totalusers")

**public** ResponseEntity<Long> getTotalUsers() {

**long** totalUsers = executiveService.getTotalUsers();

**return** ResponseEntity.*ok*(totalUsers);

}

//localhost:8081/executive/totalfeedbacks

@GetMapping("/totalfeedbacks") // Gets total number of feedbacks given

**public** ResponseEntity<Long> getTotalFeedbacks() {

**long** totalFeedbacks = executiveService.getTotalFeedbacks();

**return** ResponseEntity.*ok*(totalFeedbacks);

}