Muhammad Ibrahim 23L-0521
Maryum Nauman 23L-0696
Ali Murtaza 23L-0508
Faiq Ahsan 23L-0753

Support Staff

Identifier	UC-SupportStaff-1	
Name	View Queries	
Description	Support staff is able to view queries made by customers	
Priority	Low	
Actors	Support Staff(primary), customer(secondary)	
Pre-condition(s)		
Post-condition(s)	Queries are viewed	
Typical Course of Action		
S#	**Actor Action**	**System Response**
1	Support Staff is on his Dashboard/Home page	
2	Support Staff clicks on view Queries Button showed to him	The System returns a new page having Multiple Query objects
3	Support Staff clicks on a Query to view its Description	System responds with a Query object containing info about the Query(Description, made on, resolved or not)
Alternate Course(s) of Action		
1a	Support Staff is not logged in	A login page is displayed by the system
2a		If There are no Customer Queries Available in the system, The system Displays “No available Queries” message
2b		If the Backend/Database is not responding to Query, the system displays “404 not found” or a similar error

Identifier	UC-SupportStaff-2	
Name	Answer Queries	
Description	The support staff answers any Query	
Priority	Low	
Actors	Support Staff	
Pre-condition(s)	Queries are viewed and unresolved query must be available	
Post-condition(s)	Query(or Queries) are answered	
Typical Course of Action		
S#	**Actor Action**	**System Response**
1	Support Staff navigates to “View Queries” page	The system displays a list of pending queries
2	Support Staff selects any query	The System Shows Query Details
3	Support Staff writes a response and clicks on submit button.	The System marks the Query as Resolved and sends notification to Customer. Also redirects to the ‘View Queries” page
Alternate Course(s) of Action		
1a	Support Staff session is expired or Not logged in	The system opens The login page
2a		If There are no available Queries, The System Shows “No Queries made” message
3a		The Selected Query is Already answered, The system Disables the response input box and shows that its already resolved
3b	Support Staff just Views the query and decide not to answer it (clicks the quit icon)	The System returns to The “View Queries” page without modifying the status of the Query

Customer

Identifier	UC-Customer-1	
Name	Registration	
Description	Register the user	
Priority	Medium	
Actors	Customer	
Pre-condition(s)	User is on the registration page	
Post-condition(s)	User account gets registered	
Typical Course of Action		
S#	**Actor Action**	**System Response**
1.	Customer will enter his name in the name field	
2.	Customer will enter his date of birth from the calendar	
3.	Customer will enter the email in the email field	
4.	Customer will enter his unique username	
5.	Customer will enter the new password	
6.	Customer will enter his contact info in the contact field	
7.	Customer will click on the register button	System will redirect actor to login page
Alternate Course(s) of Action		
1(a)	Customer leaves name field empty and clicks on register button	System displays error message: “This field is required”
1(b)	Customer enters special character or number	System displays error message
3(a)	Customer leaves email field empty and clicks on register button	System displays error message: “This field is required”
3(b)		If the email format is wrong system displays error message

3(c)		If the email already exists, system displays error message
4(a)	Customer leaves username field empty and clicks on register button	System displays error message: “This field is required”
4(b)		If the username already exists, system displays error message
5(a)	Customer leaves password field empty and clicks on register button	System displays error message: “This field is required”
6(a)	Customer leaves contactinfo field empty and clicks on register button	System displays error message: “This field is required”
6(b)		If the contact does not follow a specified format, system displays an error

Identifier	UC-Customer-2	
Name	Login	
Description	Validation of the user	
Priority	Medium	
Actors	Customer	
Pre-condition(s)	Actor is registered and is on login page	
Post-condition(s)	Actor will be logged in to the system	
Typical Course of Action		
S#	**Actor Action**	**System Response**
1.	Customer shall enter username in the username field	
2.	Customer shall enter the password in the password field	
3.	Customer shall click the login button	System shall redirect the customer to homepage/dashboard
Alternate Course(s) of Action		
1(a)	Customer leaves username field empty and clicks on login page	System shall display error: “This field is required”
2(a)	Customer leaves username field empty and clicks on login page	System shall display error: “This field is required”
3(a)		If the username and password will not match then system shall display error message

Identifier	UC-Customer-3	
Name	Reset Password	
Description	Allows the user to reset his/her password	
Priority	Medium	
Actors	Customer	
Pre-condition(s)	Actor is registered and logged in	
Post-condition(s)	Actors password is renewed	
Typical Course of Action		
S#	**Actor Action**	**System Response**
1.	Customer enters registered email/username	System verifies user existence and sends otp via email
2.	Customer enters otp	System validates otp
3.	Customer enters new password	
4.	Customer enters reset button	System updates password and redirects to login page
Alternate Course(s) of Action		
1(a)		If email/username not found system displays error message “Account not found”
1(b)	Customer leaves username/email field empty and clicks reset button	System displays error message: “This field is required”
2(a)		If the otp entered is wrong display error message
3(a)	Customer leaves password field empty	System displays error message: “This field is required

Identifier	UC-Customer-4	
Name	Update Personal Profile	
Description	Allow customer to update profile information	
Priority	Medium	
Actors	Customer	
Pre-condition(s)	Customer is logged in and on update profile page	
Post-condition(s)	Profile information is updated successfully	
Typical Course of Action		
S#	**Actor Action**	**System Response**
1.	Customer navigates to update profile page	System displays existing profile data
2.	Customer updates required fields (i.e, name,email, contact etc.)	System validates input
3.	Customer clicks update button	System updates profile and shows confirmation message
Alternate Course(s) of Action		
2(a)	Customer leaves a field empty and clicks on update button	System displays error message: “This field is required”
2(b)		If email format invalid system displays error message
2(c)		If contact format invalid system displays error message
2(d)		If email already exists system displays error message

Identifier	UC-Customer-5
Name	Search Seat
Description	Allows customer to search seat based on date, time, class and destination
Priority	High
Actors	Customer
Pre-condition(s)	Customer is logged in and on the booking page
Post-condition(s)	System displays available seats on the basis of given requirements

Typical Course of Action		
S#	**Actor Action**	**System Response**
1.	Customer selects date from calendar	
2.	Customer selects time	
3.	Customer selects class	
4.	Customers select source	
5.	Customer selects destination	
6.	Customer clicks on search button	System shows available seats
Alternate Course(s) of Action		
1(a)	Customer leaves date field empty	System displays error: “This field is required”
2(a)	Customer leaves time field empty	System displays error: “This field is required”
3(a)	Customer leaves class field empty	System displays error: “This field is required”
4(a)	Customer leaves source field empty	System displays error: “This field is required”
5(a)	Customer leaves destination field empty	System displays error: “This field is required”
6(a)		If no seats available based on the requirements system shall display error

Identifier	UC-Customer-6
Name	Select Seats
Description	Allow customer to select one or more available seats
Priority	High
Actors	Customer
Pre-condition(s)	Customer has searched for available seats
Post-condition(s)	Selected seats are marked as reserved for customer

Typical Course of Action

S#	**Actor Action**	**System Response**
1.	Customer view available seats after searching	System displays available seats
2.	Customer selects one or more seats	System marks selected seats as ‘reserved’ temporarily
3.	Customers clicks confirm button	System confirms selection and proceeds to booking

Alternate Course(s) of Action

2(a)		If selected seat unavailable system displays error
2(b)	Customer selects no seat and clicks confirm	System displays error to select at least one seat

Identifier	UC-Customer-7	
Name	Book Seats	
Description	Allows customers to confirm booking of selected seats	
Priority	High	
Actors	Customer	
Pre-condition(s)	Customer has selected one or more seats	
Post-condition(s)	Seats are booked for the customer	
Typical Course of Action		
S#	**Actor Action**	**System Response**
1.	Customer reviews selected seats	System displays selected seats summary
2.	Customer clicks Book tickets	System marks seats as booked,generates unique booking id and redirects to payment process
Alternate Course(s) of Action		
1(a)		If no seats selected system displays error
2(a)		If seats sold during process system shows error: “Seats unavailable, please reselect”

Identifier	UC-Customer-8	
Name	Pay Online	
Description	Customers pay for booked tickets, with support for discount codes, vouchers, and promotional offers.	
Priority	High	
Actors	Customer	
Pre-condition(s)	Customer has booked tickets and has booking id	
Post-condition(s)	Payment is successful and tickets are confirmed	
Typical Course of Action		
S#	**Actor Action**	**System Response**
1.	Customer reviews booking and total fare	System displays final amount
2.	Customer applies discount code/vouchers/ promo	System validates and updates fares
3.	Customer enters payment details	System validates details
4.	Customer clicks Pay Now button	System processes payment and confirms booking with generating e-ticket
Alternate Course(s) of Action		
2(a)		If invalid/expired promo code system displays error
3(a)		If payment details invalid/missing system displays error
4(a)		If payment failure system displays error

Identifier	UC-Customer-9
Name	View E-ticket
Description	Allow the customer to view e-ticket generated after a successful payment.
Priority	Medium
Actors	Customer
Pre-condition(s)	If booking is successful and system has generated e-ticket
Post-condition(s)	E-ticket is displayed to customer

Typical Course of Action

S#	**Actor Action**	**System Response**
1.	Customer navigates to E-ticket section	System displays generated e-tickets

Alternate Course(s) of Action

1(a)		If no E-ticket generated, system displays error: “No e-ticket generated”

Identifier	UC-Customer-10	
Name	View Booking History	
Description	Allow customer to view history of all bookings	
Priority	Medium	
Actors	Customer	
Pre-condition(s)	Customer is logged in	
Post-condition(s)	Booking history is displayed	
Typical Course of Action		
S#	**Actor Action**	**System Response**
1.	Customer navigates to view booking history page	System retrieves booking history of the customer
2.	Customer view list of bookings	System displays bookings with details(i.e, date,time,class,etc,)
Alternate Course(s) of Action		
1(a)		If no booking history systems displays: "No booking history found"

Identifier	UC-Customer-11	
Name	Cancel Ticket	
Description	Allow customer to cancel a booked ticket if not departed	
Priority	High	
Actors	Customer	
Pre-condition(s)	Customer has an active booking and not departed yet	
Post-condition(s)	Ticket is cancelled and refund process is initiated (if applicable)	
Typical Course of Action		
S#	**Actor Action**	**System Response**
1.	Customer selects booking to cancel	System checks departure status
2.	Customer confirms cancellation	System cancels ticket and processes refund
Alternate Course(s) of Action		
1(a)		If departure has occurred system displays error: “Ticket cannot be cancelled after departure”
2(a)		If refund fails system displays error to contact support

Identifier	UC-Customer-12	
Name	View Notifications	
Description	Allow customer to view system notifications	
Priority	Medium	
Actors	Customer	
Pre-condition(s)	Customer is logged in	
Post-condition(s)	Notifications are displayed	
Typical Course of Action		
S#	**Actor Action**	**System Response**
1.	Customer clicks on notification menu	System retrieves notifications
2.	Customer view notifications	System displays booking updates, schedule changes,etc.
Alternate Course(s) of Action		
1(a)		If no notifications, system shows: “No new notifications”

Identifier	UC-Customer-13	
Name	Contact Support Staff	
Description	Allow Customers to Send queries to support staff	
Priority	Low	
Actors	Customer	
Pre-condition(s)		
Post-condition(s)	Support staff receives and may respond	
Typical Course of Action		
S#	**Actor Action**	**System Response**
1	Customer clicks on the Contact Us button on the Homepage	System Navigates to the contact Us form
2	Customer Types his issue description in the form	
3	Customer Click on the Submit Button	The System sends the Query to the Supportstaff
Alternate Course(s) of Action		
2a		If The Description Field is empty, the system displays error
3a		If customer clicks on cancel button, the system redirects the user to the home page

Admin

Identifier	UC-Admin-1
Name	Add routes or schedules
Description	Admin can add routes or schedules if not existing
Priority	High
Actors	Admin
Pre-condition(s)	Admin is logged in. Route schedule does not already exist
Post-condition(s)	New route/schedule is added

Typical Course of Action		
S#	**Actor Action**	**System Response**
1.	Admin navigates to Manage Routes/ Schedules	System shows already existing routes/ schedules
2.	Admin enters origin	
3.	Admin enters destination	
4.	Admin enters timings	
5.	Admin clicks on add route/schedule button	System adds route/ schedule with specific details
Alternate Course(s) of Action		
2(a) & 3(a)	Admin enters same origin and destination	System displays error
5(a)		If route/ schedule already exists then display error: “Route/ Schedule already exists”

Identifier	UC-Admin-2
Name	Delete routes or schedules
Description	Admin deletes an existing route or schedule.
Priority	High
Actors	Admin
Pre-condition(s)	Route/ Schedule exists in the system
Post-condition(s)	Route/ Schedule deleted

Typical Course of Action		
S#	**Actor Action**	**System Response**
1.	Admin selects route/ schedule	System shows route/ schedule details
2.	Admin clicks delete button	System remove the route/ schedule
Alternate Course(s) of Action		
1(a)		If route/ schedule does not exist system displays error

Identifier	UC-Admin-3	
Name	Update routes or schedules	
Description	Admin updates details of routes/ schedules	
Priority	High	
Actors	Admin	
Pre-condition(s)	Route/ Schedule exists	
Post-condition(s)	Route/ Schedule updated successfully	
Typical Course of Action		
S#	**Actor Action**	**System Response**
1.	Admin selects route/ schedule	System shows route/ schedule details
2.	Admin updates details	System checks for validity of details
3.	Admin clicks on update button	System updates route/ schedule
Alternate Course(s) of Action		
1(a)		If route/schedule does not exist system displays error
2(a)		If invalid update system displays error

Identifier	UC-Admin-4		
Name	Add ticket prices		
Description	Admin adds ticket prices for a route/schedule/class.		
Priority	High		
Actors	Admin		
Pre-condition(s)	At least one route/schedule exists.		
Post-condition(s)	Price added and available during booking.		
Typical Course of Action			
S#	**Actor Action**	**System Response**	
1	Admin navigates to “Ticket Prices”.	System lists current prices.	
2	Admin clicks “Add Price”.	System shows input form.	
3	Admin enters details (route, class, price).	System validates inputs.	
4	Admin Clicks on submit button.	System stores new ticket price and displays updated list.	
Alternate Course(s) of Action			
3a		If Missing or invalid values, System throws error.	
4a	Admin clicks the cancel button	The system navigates back to “Ticket prices” page without adding anything	

Identifier	UC-Admin-5	
Name	Update ticket prices	
Description	Admin changes existing ticket prices.	
Priority	High	
Actors	Admin	
Pre-condition(s)	Ticket price record exists.	
Post-condition(s)	Updated price applied to future bookings.	
Typical Course of Action		
S#	**Actor Action**	**System Response**
1	Admin selects “Manage Ticket Prices”.	System lists current prices.
2	Admin clicks “Update” on a price entry.	System displays editable form.
3	Admin modifies value (amount, class).	System validates.
4	Admin submits changes.	System updates database and shows updated record.
Alternate Course(s) of Action		
3a		If Invalid/negative price, System rejects.
4a	Admin clicks the cancel button	The system navigates back to “Ticket prices” page without updating anything

Identifier	UC-Admin-6	
Name	Delete ticket prices	
Description	Admin removes a ticket price record.	
Priority	High	
Actors	Admin	
Pre-condition(s)	Ticket price record exists.	
Post-condition(s)	Deleted record no longer available for booking.	
Typical Course of Action		
S#	**Actor Action**	**System Response**
1	Admin navigates to “Ticket Prices”.	System lists prices.
2	Admin selects entry and clicks “Delete”	System asks confirmation.
3	Admin confirms.	System removes price record and updates list
Alternate Course(s) of Action		
2a		If ticket price is currently linked to active bookings , System prevents deletion.
3a	Admin denies(click on cancel)	The system navigates back to “Ticket prices” page without deleting anything

Identifier	UC-Admin-7	
Name	Manage Seat Availability	
Description	Admin can update the status of seats (available, reserved) for a specific schedule.	
Priority	High	
Actors	Admin	
Pre-condition(s)	Route and schedule must exist	
Post-condition(s)	Seats become available for customer booking.	
Typical Course of Action		
S#	**Actor Action**	**System Response**
1	Admin navigates to “Seat Availability”	System displays current seat layout.
2	Admin selects seats to change status (available, reserved).	System validates seat selection.
3	Admin confirms action.	System updates seat status in database and System refreshes and displays updated seat map.
Alternate Course(s) of Action		
2a		If seat capacity is exceeded , System rejects addition.
3a	Admin denies (clicks on cancel)	The system navigates back to “seat availability” page

Identifier	UC-Admin-8		
Name	Manage Booking Statistics and Reports		
Description	Admin can generate and view booking statistics, trends, and revenue reports.		
Priority	Medium		
Actors	Admin		
Pre-condition(s)	Admin logged in; booking data exists.		
Post-condition(s)	Reports are generated and available for viewing/export.		
Typical Course of Action			
S#	**Actor Action**	**System Response**	
1	Admin opens “Reports & Statistics” section.	System displays available report options.	
2	Admin selects report type (daily, monthly, revenue, bookings by route, etc.).	System fetches required data.	
3	Admin requests generation of the report	System processes and generates report.	
Alternate Course(s) of Action			
3a		If no booking data found for selected filter , System shows “No Data Available”	

Identifier	UC-Admin-9		
Name	Set Cancellation and Refund Policies		
Description	Admin defines or updates cancellation and refund rules applied to bookings.		
Priority	High		
Actors	Admin		
Pre-condition(s)	Admin logged in.		
Post-condition(s)	Policies are saved and applied to all future bookings.		
Typical Course of Action			
S#	**Actor Action**	**System Response**	
1	Admin navigates to “Policies Management”.	System shows existing cancellation/refund policies.	
2	Admin adds or updates rules (e.g., refund % if cancelled before X hours).	System validates input	
3	Admin confirms changes.	System saves new policies in database	
Alternate Course(s) of Action			
2a		If input is invalid (e.g., refund > 100%) , System rejects and prompts correction.	
3a	Admin clicks cancel	The system will navigate back to the manage cancellation policies page.	

Identifier	UC-Admin-10	
Name	Add Promotional Codes and Discounts	
Description	The admin can create promotional codes and special discounts to encourage bookings.	
Priority	medium	
Actors	Admin	
Pre-condition(s)	Admin logged in	
Post-condition(s)	New Promo codes added	
Typical Course of Action		
S#	**Actor Action**	**System Response**
1	Admin click on Manage promocodes option in his Dashboard	System displays a new page showing existing promo codes along with operations that can be performed
2	Admin clicks on Add promo code and fills out necessary info required in its field(i.e. Discount percentage, max cap)	
3	Admin Saves the promo code	The system adds the promo code to database and updates the page
Alternate Course(s) of Action		
2a		If any required field (percentage, discount cap) is left blank, system throws an error
3a	Admin clicks cancel	The system will navigate back to the manage promo codes page

Identifier	UC-Admin-11	
Name	Update promo code	
Description	The admin can edit out the fields of an existing promo code	
Priority	Medium	
Actors	Admin	
Pre-condition(s)	Admin is logged in	
Post-condition(s)	Promo codes are modified	
Typical Course of Action		
S#	**Actor Action**	**System Response**
1	Admin click on Manage promocodes option in his Dashboard	System displays a new page showing existing promo codes along with operations that can be performed
2	Admin clicks on edit button on one object of promo code	The system shows a popup, showing input fields that are already filled with previous data
3	Admin changes any field	
4	Admin clicks on save button	The Promo code is modified
Alternate Course(s) of Action		
3a		If any field is left empty, the system throws error
4a	Admin clicks on cancel button	The system will navigate back to the manage promo codes page without editing anything

Identifier	UC-Admin-12	
Name	Delete promo code	
Description	The admin can delete an existing promo code	
Priority	Medium	
Actors	Admin	
Pre-condition(s)	Admin is logged in	
Post-condition(s)	Promo codes are removed from database	
Typical Course of Action		
S#	**Actor Action**	**System Response**
1	Admin click on Manage promocodes option in his Dashboard	System displays a new page showing existing promo codes along with operations that can be performed
2	Admin clicks on delete button on one object of promo code	The system shows a popup, Asking “are you sure?”
3	Admin clicks “Yes”	The system deletes that promo code from the database and returns to the updated “manage promocodes” page
Alternate Course(s) of Action		
3a	Admin clicks on cancel button	The system will navigate back to the manage promo codes page without deleting anything