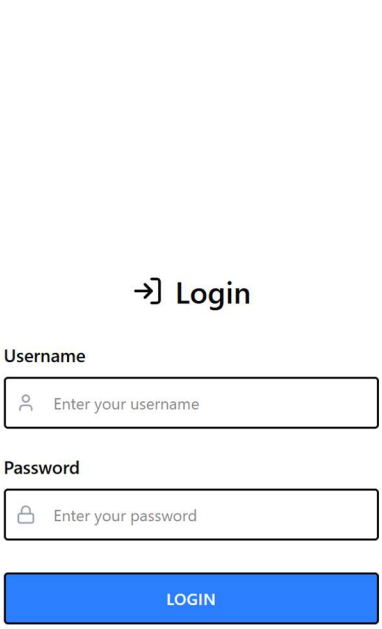
Dec 26, 2025

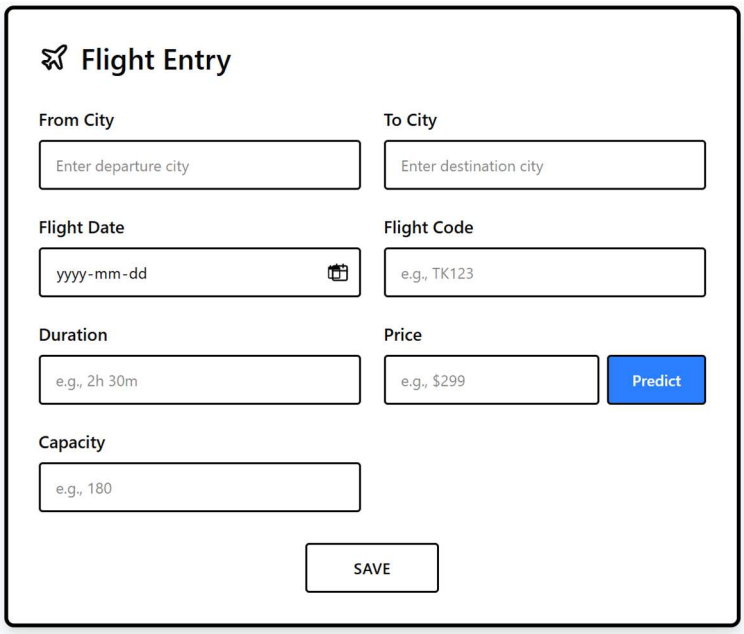
SE 4458 Software Architecture & Design of Modern Large Scale Systems - Final

Group 1 : Airline Ticketing System

Create an airline ticketing system like turkishairlines.com . Implement Below are use cases for functional and non-functional requirements.

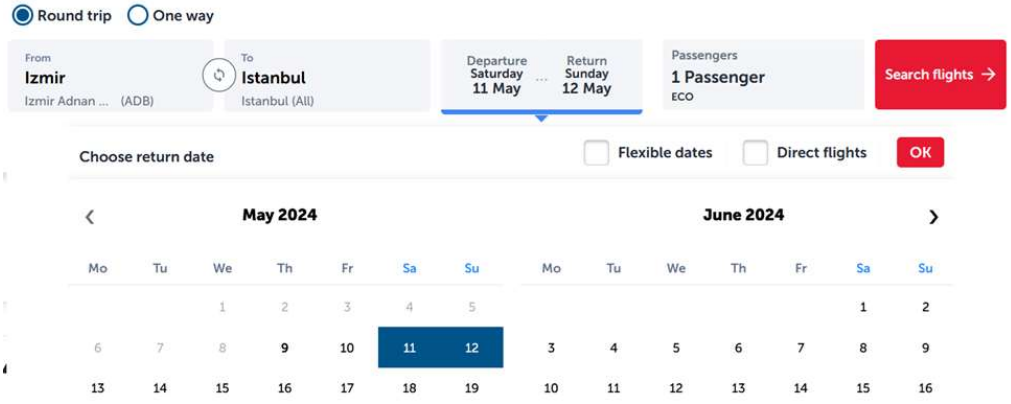
ADD FLIGHTS

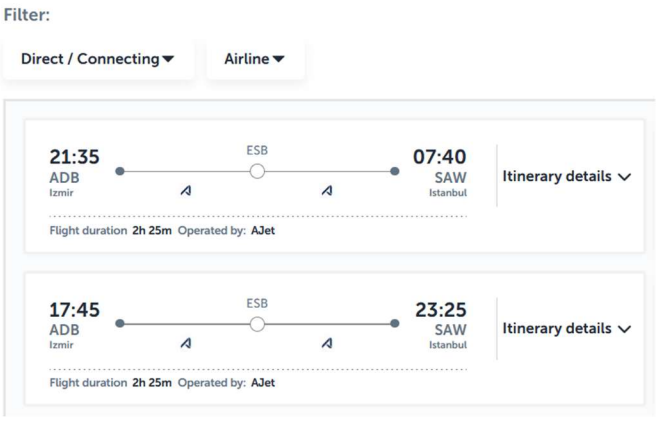




- Authenticated Admin users can add flights to specified dates with capacity. This will be entered from a separate domain like admin.airlines.com. Users who can use this screen will be in ADMIN role
- For price prediction, you will build a simple price prediction model using Machine Learning. You can use a dataset like https://www.kaggle.com/datasets/shubhambathwal/flight-price-prediction . This will predict the price of flight based on duration, time (and other attribute you wish based on dataset you work on)

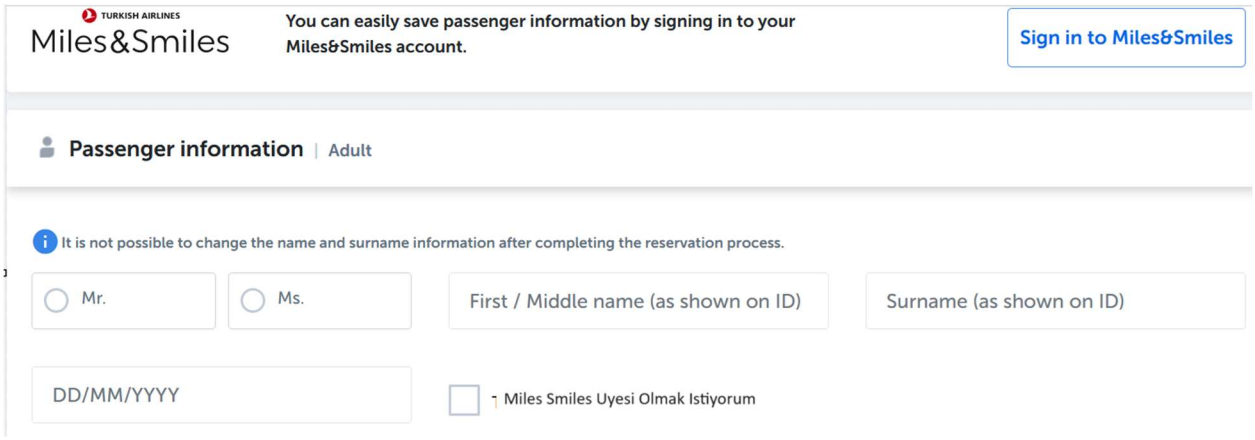
SEARCH FLIGHTS





- Users will be able to search by airport, dates and number of passengers
- Flexible dates and direct flight options should be available
-

BUY TICKET

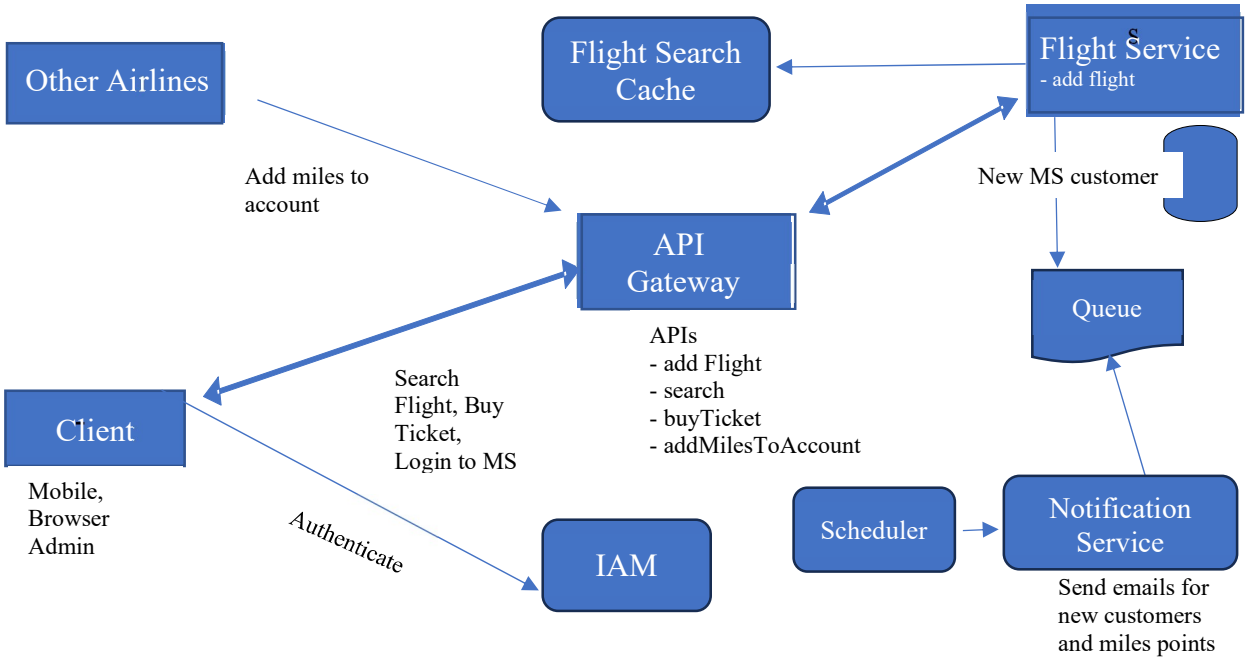


- When buying a ticket, capacity in the flight must be reduced as many as number of passengers
- If a passenger is a MilesSmiles member
    - they can login and their information will be populated. Moreover, purchases will be saved with MilesSmiles member number
    - they can buy the ticket with MilesSmiles points, if they have enough of them
- if passenger wants to be a MilesSmiles member
    - create a profile and later send a welcome email
- NO transaction needed for payment

ADD MILES TO MILES&SMILES ACCOUNT

- once the flights of MilesSmiles customers end, a nightly process runs and updates points in member accounts
- other airlines that are in MilesSmiles program can also update miles points. THIS WILL BE AN AUTHENTICATED SERVICE
- write a scheduled task
    - to go over flights and members on the run date and send emails if there are any points added to account balance
    - to pull new customers from the queue and send them a welcome message
- For email send, you can use a gmail account. see https://kinsta.com/blog/gmail-smtp-server/

NON-FUNCTIONAL REQUIREMENTS



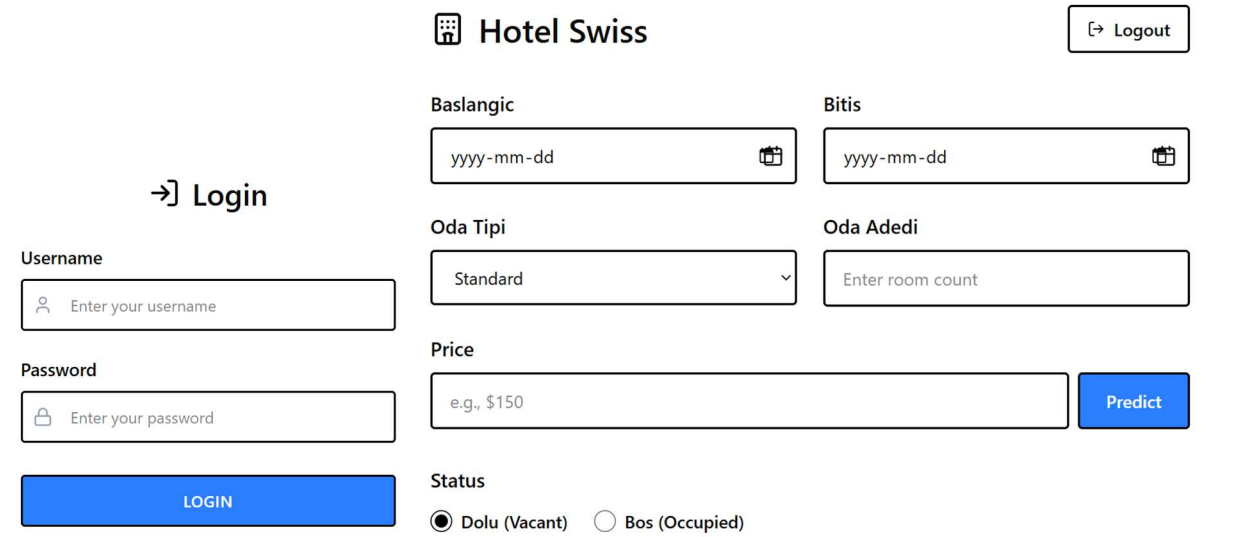
Students

DOGUKAN YESILKAYA	
IRMAK ARABACI	KAAN YILMAZ
AYSIMA ADATEPE	ULKU BARTU SERBEST
MELISA DEMIRBAS	ALI HAKTAN SIGIN
HUSEYIN BALCI	BASAR OZKASLI
KEREM KOYUNCU	MELİSA ŞENER
MELIKE AYTAC	LARA ÖZDUMAN
YAGMUR SABIRLI	DİLA GENÇAĞA
NURETTIN DEMIREL	CEMİL FAHRECİ

Group 2: Create a Hotel Booking System

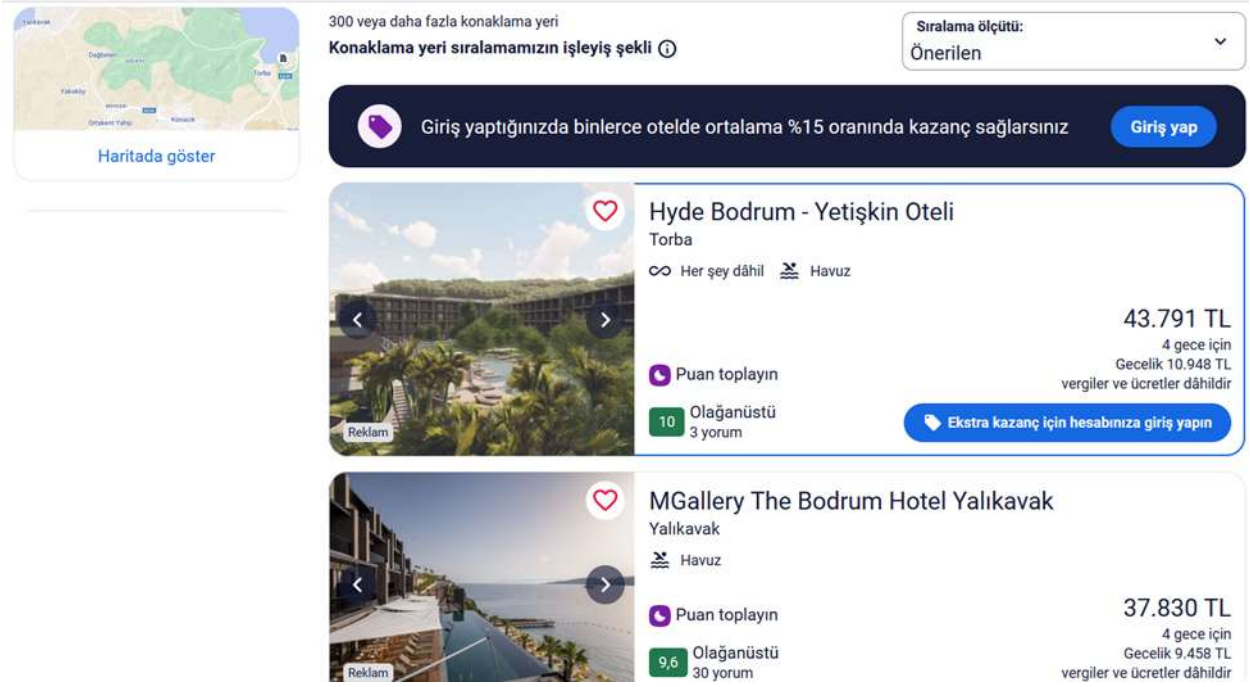
Create a hotel booking system like Hotels.com. Implement Below are use cases for functional and non-functional requirements.

HOTEL ADMIN SERVICE



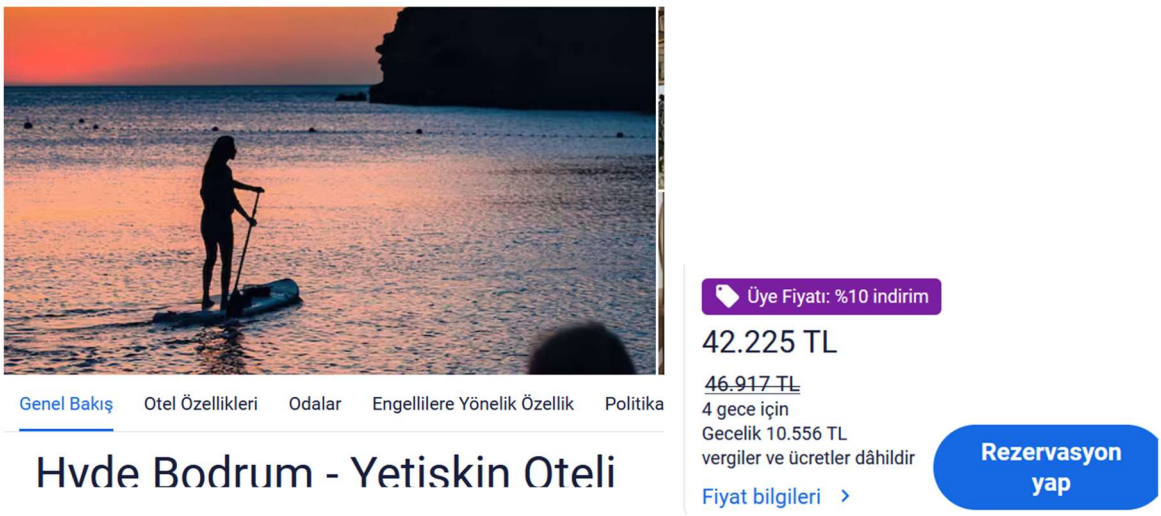
- Admins for hotels can add/update rooms for availability between start and end dates. This will be entered from a separate domain like admin.hotels.com. Users who can use this screen will be in ADMIN role
- For price prediction, you will build a simple price prediction model using Machine Learning. You can use a dataset like https://www.kaggle.com/datasets/maelysboudier/hotel-prices-in-europe . This will predict the price of hotel room based on time of the year, size (and other attribute you wish based on dataset you work on)

HOTEL SEARCH SERVICE



- User can search hotels by destination point, dates and number of people. Only rooms that admins specified as vacant will appear in results for given dates
- Client who login to application will see 10% discounted prices.
- ‘Haritada goster’ is required to show hotels that have been searched

BOOK HOTEL SERVICE

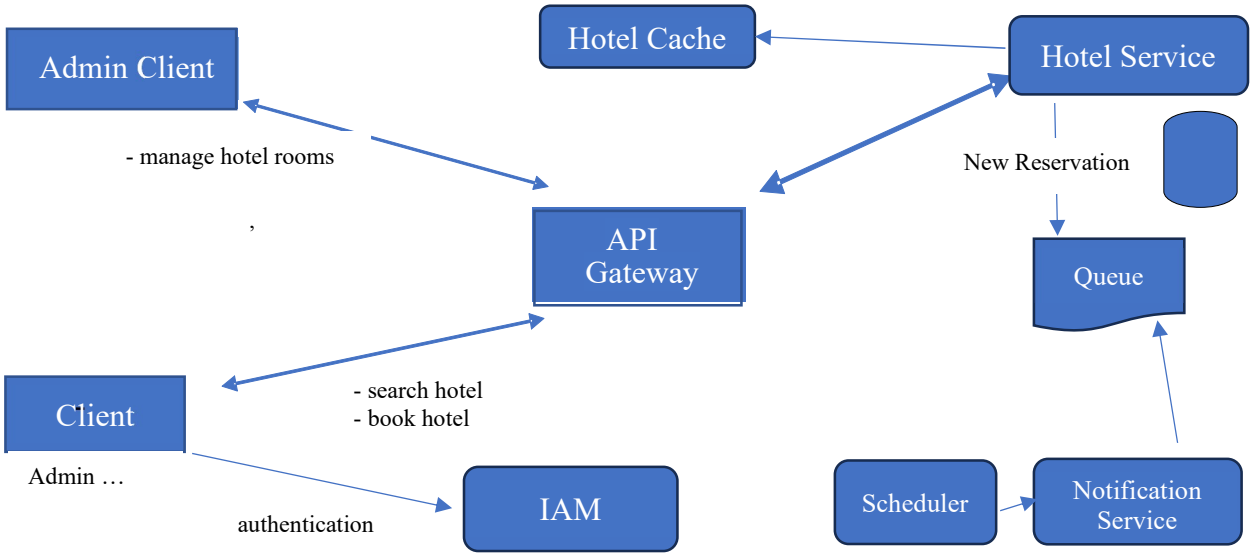


- On hotel detail page, user can book the hotel. Capacity of the hotel needs to be decreased for specified dates
- NO transaction needed for payment

NOTIFICATION SERVICE

- write a nightly scheduled task
    - to go over all hotel capacities and notify hotel administrators when it is below 20% for the next month.
    - to pull new hotel reservations from the queue and send them a message about reservation details

NON-FUNCTIONAL REQUIREMENTS



Students

ELİF EMİNE GÜNAL	OZAN BÖCE
BERKAY HEREK	MURAT HABİP OKAN
MEHMET UTKU GÜNDOĞDU	PELİN DUMAN
EMRE ŞENER	SELÇUK SUAT SAYIN
ESRA ECE GÜNGÜNEY	DEFNE TEKYİĞİT
DİLARA ACAR	DURU GENCAY
AHMET KEMAL BİLİCİLER	SUDE KARAKAYA
	TEVFİK EFE AYDIN
	DENIZ YALIM YILMAZ

COMMON REQUIREMENTS

- Project can be built on any service-oriented framework as long as requirements are met.
- Simple UI implementation per mock-ups given above **is required**. Front end UIs do not have to be same as shown above. It just needs to work
- All business use cases above must be available via REST webservices
- The project will be deployed to providers like Azure, Google Cloud and AWS or any other cloud vendor. Deployment is BONUS (+20 points)
- Per deployment diagrams above, Flight, Hotel and Notification services will be deployed separately. All APIs will be reached via an API gateway
- All user authentication will be stored in an IAM service like AWS Cognito. No external authentication like Google needed. You local authentication implementation will not be accepted
- You can use RabbitMQ, AWS SQS or Azure Messaging for queue solutions. If you cant implement it, you will need to implement it synchronously but you will lose points
- For caching, you can use a distributed cache like REDIS or memory cache
- All REST services must be versionable and support pagination when needed
- You can choose any development environment you like as long as they support REST services and is deployable in cloud.
- You can make assumptions as long as you document them
- At least one caching solution needs to be implemented i.e Airport names , Airline destinations
- MAKE SURE TO CREATE A DOCKER FILE in your source. See an example at https://www.digitalocean.com/community/tutorials/how-to-build-a-node-js-application-with-docker for a node.js application. DO NOT CREATE A DOCKER IMAGE FILE, it would be too big and not needed
- DEPLOYMENT
    - create a data model for each data source and use a DATABASE service from any cloud service you like i.e Azure SQL Server, Azure Cosmos Table. SQLite is NOT allowed
    - For API and UI layer hosting of app services, use a cloud service i.e Azure App Services, AWS App runner
    - For API gateway, you can use the gateway you implemented in assignment. AWS API gateway is free. Azure API management is costly so refrain from using it unless you want to pay for it
    - For queue service, use Azure Message Queues, Rabbit MQ
    - For scheduling services, use a cloud service i.e AWS Scheduler, Azure App Logic or https://cloud.google.com/scheduler/

DELIVERABLES

- Public Github repository link
- A README document in your GITHUB repository that has
    - Final deployed urls of the application

- your design, assumptions, and issues you encountered.
- Data models (i.e an ER)
- Include a link to a short video (max 5 minutes) presenting your project.