BankFlow Challenge 2026

۱. مقدمه

در این چالش شما مسئول طراحی و پیاده‌سازی یک سامانه Backend برای پردازش درخواست‌های تسهیلات بانکی خواهید بود. هدف این چالش صرفاً نوشتن چند API یا پیاده‌سازی یک سری قابلیت مشخص نیست؛ بلکه انتظار می‌رود راهکاری ارائه کنید که از نظر طراحی نرم‌افزار، قابلیت توسعه، خوانایی کد و نگهداری، کیفیت مناسبی داشته باشد.در این مسابقه علاوه بر صحت عملکرد سیستم، کیفیت تصمیم‌های مهندسی شما نیز مورد ارزیابی قرار خواهد گرفت.
بنابراین معماری سیستم، نحوه مدل‌سازی فرآیندها، تفکیک مسئولیت‌ها و قابلیت توسعه راهکار، به اندازه عملکرد صحیح آن اهمیت دارد.

۲. داستان مسئله

شرکت داتین یکی از ارائه‌دهندگان زیرساخت‌های بانکی (Banking Service Provider) است و سرویس‌های مختلفی را در اختیار بانک‌ها و مؤسسات مالی قرار می‌دهد. یکی از محصولات این شرکت، سامانه پردازش درخواست تسهیلات است.روزانه هزاران درخواست وام از بانک‌های مختلف وارد این سامانه می‌شود.هر درخواست باید قبل از تأیید نهایی، مراحل مختلفی را طی کند.

به عنوان نمونه:

ثبت درخواست

↓

اعتبارسنجی اولیه

↓

بررسی تقلب

↓

اعتبارسنجی مالی

↓

تأیید مدیر

↓

تأیید نهایی

اما تمام بانک‌ها فرآیند یکسانی ندارند.
برای مثال:
بانک اول

اعتبارسنجی

↓

بررسی تقلب

↓

اعتبارسنجی مالی

↓

تأیید

بانک دوم

اعتبارسنجی

↓

بررسیAML

↓

بررسی تقلب

↓

بررسی ضامن

↓

اعتبارسنجی مالی

↓

تأیید

بانک سوم

اعتبارسنجی

↓

تأیید مدیر

↓

تأیید

مدیریت شرکت تصمیم گرفته است نسخه دوم این سامانه را با تمرکز بر توسعه‌پذیری و نگهداری آسان طراحی کند.
شما به عنوان تیم توسعه مسئول طراحی این نسخه جدید هستید.

.۳هدف چالش

هدف این چالش طراحی و پیاده‌سازی یک **سامانه پردازش درخواست تسهیلات** است که بتواند فرآیند بررسی درخواست‌های وام را مدیریت کند.

سامانه باید امکان موارد زیر را فراهم کند:

-ثبت درخواست تسهیلات
-پردازش درخواست بر اساسWorkflow
-مشاهده وضعیت فعلی درخواست
-مشاهده تاریخچه پردازش درخواست

در این چالش تنها پیاده‌سازی قابلیت‌ها ملاک ارزیابی نیست.

انتظار می‌رود راهکار ارائه شده دارای ویژگی‌های زیر باشد:

-طراحی مناسب
-قابلیت توسعه
-قابلیت نگهداری
-خوانایی مناسب
-قابلیت تست
-تفکیک صحیح مسئولیت‌ها

.۴محدوده مسئله

در این مسابقه تنها مسئولیت شما پیاده‌سازی **سامانه پردازش درخواست تسهیلات** است.

تمام سرویس‌ها و قابلیت‌هایی که خارج از این محدوده قرار دارند، جزئی از این چالش محسوب نمی‌شوند.

##قابلیت‌های الزامی

راهکار شما باید حداقل قابلیت‌های زیر را پیاده‌سازی کند:

-ثبت درخواست تسهیلات
-اجرای فرآیند بررسی درخواست
-مدیریت وضعیت درخواست
-ثبت تاریخچه اجرای مراحل
-نگهداری اطلاعات پس ازRestart
-ارائهAPI هایREST
-اجرای پروژه از طریقDocker

##موارد خارج از محدوده

پیاده‌سازی موارد زیر الزامی نیست:

-احراز هویت کاربران
-مدیریت نقش‌ها و سطح دسترسی
-رابط کاربری(Frontend)
-اتصال واقعی به سامانه‌های بانکی
-ارسال پیامک
-ارسال ایمیل
- Kubernetes
-استقرار رویCloud
- CI/CD
-معماریMicroservice
- Kafka، RabbitMQ -یا سایرMessage Broker ها
-پردازش توزیع‌شده

در صورت استفاده از این فناوری‌ها امتیاز اضافه‌ای تعلق نخواهد گرفت.

.۵نیازمندی‌های عملکردی

سامانه باید قابلیت‌های زیر را ارائه دهد.

FR-1 ##ثبت درخواست تسهیلات

کاربر باید بتواند یک درخواست جدید ثبت کند.

برای هر درخواست، سامانه باید یک شناسه یکتا تولید کند.

پس از ثبت موفق، وضعیت اولیه درخواست برابر خواهد بود با:

SUBMITTED

و اولین مرحله پردازش:

VALIDATION

FR-2 ##پردازش درخواست

سامانه باید فرآیند بررسی درخواست را اجرا کند.

اجرای فرآیند تا زمانی ادامه پیدا می‌کند که درخواست به یکی از وضعیت‌های نهایی زیر برسد:

APPROVED یا REJECTED یا MANUAL_REVIEW

FR-3 ##مشاهده اطلاعات درخواست

کاربر باید بتواند اطلاعات فعلی یک درخواست را مشاهده کند.

حداقل اطلاعات قابل نمایش عبارت‌اند از:

-شناسه درخواست
-اطلاعات ثبت‌شده
-وضعیت فعلی
-مرحله فعلی
-زمان ایجاد
-زمان آخرین تغییر

FR-4 ##مشاهده تاریخچه پردازش

کاربر باید بتواند تاریخچه اجرای مراحل مختلف را مشاهده کند.

برای هر مرحله اجرا شده، حداقل اطلاعات زیر باید ذخیره شود:

-نام مرحله
-نتیجه اجرا
-زمان اجرا
-علت یا توضیح نتیجه

تاریخچه باید به ترتیب زمانی نمایش داده شود.

FR-5 ##اجرایWorkflow

هر درخواست باید از چند مرحله پردازش عبور کند.

هر مرحله تنها مسئول انجام یک وظیفه مشخص است.

برای مثال:

مرحله اعتبارسنجی تنها وظیفه بررسی صحت اطلاعات ورودی را دارد و نباید مسئول بررسی اعتبار مالی یا تشخیص تقلب باشد.

FR-6 ##مدیریت وضعیت

هر درخواست در هر لحظه تنها یک وضعیت جاری دارد.

تغییر وضعیت‌ها باید قابل پیش‌بینی و مطابق قوانین Workflow باشد.

وجود چند وضعیت همزمان برای یک درخواست مجاز نیست.

FR-7 ##ماندگاری اطلاعات

اطلاعات درخواست‌ها باید پس از Restart شدن برنامه نیز حفظ شوند.

حداقل اطلاعات زیر باید ذخیره شوند:

-اطلاعات درخواست
-وضعیت فعلی
-مرحله فعلی
-تاریخچه اجرای مراحل

روش ذخیره‌سازی اطلاعات بر عهده شرکت‌کننده است.

استفاده صرف از حافظه (In-Memory Storage) این نیازمندی را برآورده نمی‌کند.

FR-8 ##بازیابی پس ازRestart

در صورت راه‌اندازی مجدد برنامه، درخواست‌هایی که قبلاً ثبت شده‌اند باید همچنان قابل مشاهده و پردازش باشند.

Restartشدن برنامه نباید باعث از بین رفتن اطلاعات شود.

FR-9 ##جلوگیری از پردازش تکراری

اگر عملیات پردازش یک درخواست بیش از یک بار اجرا شود، سیستم نباید:

-مراحل قبلی را مجدداً اجرا کند.
-تاریخچه تکراری ایجاد کند.
-وضعیت نهایی درخواست را تغییر دهد.

FR-10 ##توسعه‌پذیری

یکی از مهم‌ترین اهداف این چالش، طراحی سیستمی است که در آینده بتوان به سادگی مراحل یا قوانین جدیدی به آن اضافه کرد.

فرض کنید بانک در آینده بخواهد:

-مرحله جدیدی به Workflow اضافه کند.
-ترتیب اجرای مراحل را تغییر دهد.
-قوانین تصمیم‌گیری جدید تعریف کند.

معماری سامانه باید به گونه‌ای باشد که اعمال این تغییرات نیازمند بازنویسی گسترده سیستم نباشد.

پیاده‌سازی این قابلیت‌ها در این مرحله الزامی نیست؛ اما طراحی باید از چنین توسعه‌هایی پشتیبانی کند.

۶. #انتظارات کیفی

راهکار ارائه شده علاوه بر صحت عملکرد، باید از نظر کیفیت پیاده‌سازی نیز قابل قبول باشد.

در طراحی سامانه انتظار می‌رود موارد زیر رعایت شوند:

-خوانایی مناسب کد
-تفکیک مسئولیت‌ها
-وابستگی کم بین بخش‌های مختلف
-قابلیت تست

-قابلیت توسعه
-قابلیت نگهداری

استفاده ازDesign Pattern ها اجباری نیست.

استفاده از یک Pattern تنها زمانی ارزشمند است که واقعاً مشکلی از طراحی را حل کند.

پیچیدگی بیشتر، به معنی کیفیت بالاتر نیست.

.۷ #آزادی در انتخاب فناوری

شرکت‌کنندگان در انتخاب زبان برنامه‌نویسی ،Framework ،پایگاه داده و ابزارهای توسعه کاملاً آزاد هستند.

استفاده از فناوری خاص، امتیاز مثبت یا منفی ایجاد نخواهد کرد.

کیفیت راهکار مهم‌تر از فناوری انتخاب شده است.

.۸اقلام قابل تحویل

هر تیم باید موارد زیر را تحویل دهد:

هدف	فایل
پیاده‌سازی	`Source Code`
اجرای پروژه	`Dockerfile`
راهنمای اجرا	`README.md`
معرفی معماری	`DESIGN.md`
تحلیل و تصمیم‌های مهندسی	`ENGINEERING_DECISIONS.md`
استراتژی و پوشش تست‌ها	`TESTING.md`

در صورت نیاز می‌توانید فایل‌های تکمیلی نیز اضافه کنید.

.۹نکات پایانی

این چالش به گونه‌ای طراحی شده است که بیش از آنکه یک مسئله الگوریتمی باشد، یک مسئله مهندسی نرم‌افزار است.

راه‌حل واحدی برای این مسئله وجود ندارد.

شرکت‌کنندگان باید با تحلیل مسئله، مناسب‌ترین طراحی را انتخاب کرده و بتوانند از تصمیم‌های مهندسی خود دفاع کنند.

.۱۰مدل دامنه(Domain Model)

در این چالش، هر درخواست تسهیلات یک موجودیت (Entity) مستقل محسوب می‌شود که در طول چرخه عمر خود، از مراحل مختلف Workflow عبور می‌کند.

هر درخواست حداقل شامل اطلاعات زیر است.

فیلد	نوع	توضیح
loanId	String	شناسه یکتای درخواست (توسط سیستم تولید می‌شود)
customerId	String	شناسه مشتری
amount	Integer	مبلغ درخواست
phone	String	شماره تلفن همراه
loanType	Enum	نوع تسهیلات
monthlyIncome	Integer	درآمد ماهانه
creditScore	Integer	امتیاز اعتباری
hasGuarantor	Boolean	وجود یا عدم وجود ضامن
status	Enum	وضعیت فعلی درخواست
currentStage	Enum |مرحله فعلی| Workflow
createdAt	Timestamp	زمان ایجاد
updatedAt	Timestamp	آخرین زمان تغییر

##انواع تسهیلات

در این نسخه تنها دو نوع تسهیلات تعریف شده است.

PERSONAL و BUSINESS

در آینده ممکن است انواع دیگری نیز اضافه شوند.

۱۱.وضعیت‌های سیستم(Loan Status)

هر درخواست در هر لحظه دقیقاً یکی از وضعیت‌های زیر را دارد.

وضعیت	توضیح
SUBMITTED |درخواست ثبت شده است| .
IN_PROGRESS |درخواست در حال پردازش است| .
MANUAL_REVIEW |نیاز به بررسی دستی دارد| .
APPROVED |درخواست تأیید شده است| .
REJECTED |درخواست رد شده است| .

پس از ورود به وضعیت‌های APPROVED یا ،REJECTED پردازش خاتمه می‌یابد.

۱۲.مراحلWorkflow

سیستم باید حداقل مراحل زیر را پشتیبانی کند.

مرحله	مسئولیت
VALIDATION	بررسی صحت اطلاعات ورودی
FRAUD_CHECK	بررسی تقلب
GUARANTOR_CHECK	بررسی وجود ضامن
CREDIT_CHECK	بررسی اعتبار مالی
MANAGER_APPROVAL	تأیید مدیر
MANUAL_REVIEW	بررسی دستی

هر مرحله فقط مسئول یک وظیفه است.

منطق دو مرحله نباید داخل یک کلاس یا ماژول پیاده‌سازی شود.

۱۳.نتیجه اجرای هر مرحله

هر مرحله تنها یکی از نتایج زیر را تولید می‌کند.

PASS

FAIL

MANUAL_REVIEW

در صورت FAIL پردازش متوقف شده و وضعیت نهایی برابر REJECTED خواهد بود.

در صورت MANUAL_REVIEW درخواست وارد وضعیت MANUAL_REVIEW می‌شود.

۱۴. Workflow پایه

تمام درخواست‌ها از مسیر زیر آغاز می‌شوند.

SUBMITTED

↓

VALIDATION

اگر VALIDATION موفق باشد

↓

FRAUD_CHECK

در غیر این صورت

↓
REJECTED

.۱۵قوانین اعتبارسنجی(Validation)

مرحله VALIDATION حداقل باید قوانین زیر را بررسی کند.

##شناسه مشتری

الزامی است.

نباید خالی باشد.

##مبلغ

باید بزرگـتر از صفر باشد.

در غیر این صورت

INVALID_AMOUNT

##شماره تلفن

الزامی است.

باید:

۱۱ -رقم باشد.
-با 09 شروع شود.
-فقط شامل رقم باشد.

در غیر این صورت

INVALID_PHONE

##نوع تسهیلات

فقط یکی از مقادیر زیر مجاز است.

PERSONAL

BUSINESS

##درآمد ماهانه

نباید منفی باشد.

##امتیاز اعتباری

باید بین 0 تا 1000 باشد

اگر هر یک از قوانین بالا نقض شود، نتیجه مرحله VALIDATION برابر FAIL خواهد بود.

۱۶.قوانین بررسی تقلب(Fraud Check)

به منظور جلوگیری از وابستگی به سرویس‌های خارجی، رفتار این مرحله به صورت Mock تعریف شده است.

اگر customerId با عبارت زیر شروع شود

FRAUD-

نتیجه مرحله

FAIL

خواهد بود.

اگر customerId با عبارت

REVIEW-

شروع شود

نتیجه

MANUAL_REVIEW

خواهد بود.

در سایر موارد

PASS

۱۷.مسیرهای شرطیWorkflow

Workflowاین سامانه خطی نیست.

مسیر پردازش بر اساس اطلاعات درخواست تغییر می‌کند.

##درخواستPERSONAL

VALIDATION
↓

FRAUD_CHECK

↓

CREDIT_CHECK

↓

APPROVED

در صورتی که مبلغ از حد مشخصی بیشتر باشد

↓

MANAGER_APPROVAL

##درخواستBUSINESS

VALIDATION

↓

FRAUD_CHECK

↓

GUARANTOR_CHECK

↓

CREDIT_CHECK

↓

APPROVED

۱۸.بررسی ضامن

این مرحله فقط برای وام‌های BUSINESS اجرا می‌شود.

اگر

hasGuarantor == false

نتیجه مرحله FAIL خواهد بود.در غیر این صورت PASS

۱۹.اعتبارسنجی مالی

این مرحله بر اساس creditScore تصمیم می‌گیرد.

اگر

creditScore < 500

↓

FAIL

اگر

500 <= creditScore < 650

↓

MANUAL_REVIEW

اگر

creditScore >= 650

↓

PASS

۲۰. تأیید مدیر

اگر مبلغ درخواست از حد تعیین شده بیشتر باشد، مرحله تأیید مدیر اجرا می‌شود.

در این نسخه مقدار پیش‌فرض این حد برابر است با 500000000

این مقدار باید قابل تغییر باشد.

رفتار Mock مدیر

اگر

amount > monthlyIncome × 20

↓

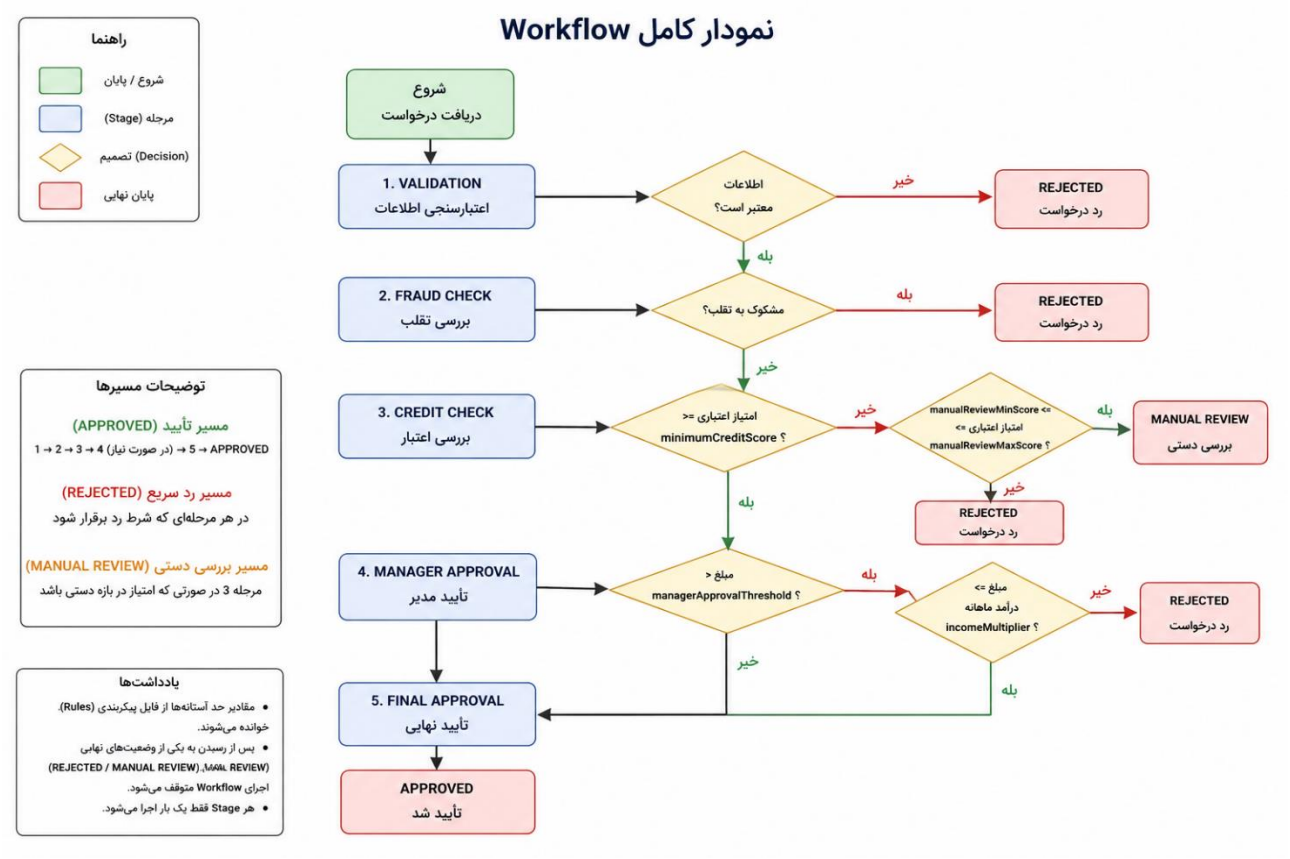
FAIL

در غیر این صورت

↓

PASS

۲۱.نمودار کاملWorkflow



۲۲.قابلیت توسعه

در طراحی سیستم فرض کنید در آینده بانک اعلام کند:

AML_CHECK بعد از FRAUD_CHECK اجرا شود. یا LEGAL_CHECK برای وام‌های BUSINESS اضافه شود. یا RISK_SCORE قبل از CREDIT_CHECK اجرا شود.

پیاده‌سازی این مراحل در نسخه فعلی الزامی نیست.

اما معماری سیستم باید به گونه‌ای باشد که اضافه شدن چنین مرحله‌هایی نیازمند بازنویسی گسترده منطق اصلی سیستم نباشد.

۲۳.رفتار مورد انتظار در بررسی دستی

اگر هر مرحله نتیجه MANUAL_REVIEW برگرداند:

-پردازش متوقف می‌شود.
-وضعیت درخواست برابر MANUAL_REVIEW خواهد شد.
-مرحله فعلی خاتمه می‌یابد.
-اجرای مجدد Workflow نباید بدون تصمیم جدید ادامه پیدا کند.

در این نسخه پیاده‌سازی سیستم بررسی دستی الزامی نیست.

۲۴. پیکربندی قوانین کسب‌وکار (Business Rules Configuration)

یکی از اهداف این چالش، طراحی سامانه‌ای است که تغییر قوانین کسب‌وکار بدون نیاز به تغییر کدهای اصلی امکان‌پذیر باشد.

به همین منظور، بخشی از قوانین سیستم باید از طریق فایل پیکربندی قابل تنظیم باشند.

پیاده‌سازی این فایل می‌تواند با هر فرمتی انجام شود؛ برای مثال:

- JSON
- YAML
- TOML
- Properties
-هر فرمت مشابه دیگر

فرمت فایل به انتخاب تیم است.

##حداقل قوانین قابل پیکربندی

سامانه باید حداقل از تنظیمات زیر پشتیبانی کند.

###حداقل امتیاز اعتباری قابل قبول

minimumCreditScore

نمونه:

```
json
{
  "minimumCreditScore": 650
}
```

در صورت تغییر مقدار این پارامتر، سیستم نباید نیازمند تغییر کد باشد.

###سقف نیاز به تأیید مدیر

managerApprovalThreshold

نمونه

```
json
{
  "managerApprovalThreshold": 500000000
}
```

در صورت تغییر این مقدار، مسیر Workflow باید به صورت خودکار تغییر کند.

###ضریب بررسی درآمد

در مرحله Manager Approval رابطه زیر استفاده می‌شود:

amount > monthlyIncome × incomeMultiplier

مقدار ضریب باید قابل تنظیم باشد.

نمونه:

```
json
{
   "incomeMultiplier": 20
}
```

###حداقل امتیاز برای بررسی دستی

نمونه:

```
json
{
   "manualReviewMinScore": 500,
   "manualReviewMaxScore": 649
}
```

در صورت تغییر این بازه، رفتار سیستم باید بدون تغییر کد اصلاح شود.

##نکات

تمام مقادیر فوق باید در زمان راه‌اندازی برنامه خوانده شوند.

در این نسخه نیازی به Reload شدن فایل در زمان اجرا وجود ندارد.

.۲۵فایل نمونه

نمونه فایل

rules.json

```
json
{
  "minimumCreditScore":650,

  "managerApprovalThreshold":500000000,

  "incomeMultiplier":20,

  "manualReview":{
     "minScore":500,
```

```
    "maxScore":649
  }
}
```

۲۶. Requirement جدید

منطق سیستم نباید شامل اعداد ثابت (Magic Numbers) باشد.

برای مثال موارد زیر نباید مستقیماً در کد نوشته شوند.

java
```
if(score>=650)
```

یا

java
```
if(amount>500000000)
```

یا

java
```
if(score>=500 && score<650)
```

این مقادیر باید از فایل پیکربندی خوانده شوند.

۲۷. قابلیت توسعه

فرض کنید بانک در آینده فایل زیر را ارسال کند.

json
```
{
  "minimumCreditScore":700,

  "managerApprovalThreshold":900000000,

  "incomeMultiplier":15,

  "manualReview":{
    "minScore":650,
    "maxScore":699
  }
}
```

در این حالت انتظار می‌رود رفتار سامانه بدون تغییر کد و تنها با تغییر فایل پیکربندی اصلاح شود.

۲۸. محدودیت

هدف این بخش پیاده‌سازی یک Rule Engine عمومی نیست.

شرکت‌کنندگان تنها کافی است قوانین تعریف‌شده در این مستند را از فایل پیکربندی بخوانند و در تصمیم‌گیری سیستم استفاده کنند.

پیاده‌سازی Rule Language ،DSL یا Rule Engine اختصاصی امتیاز اضافه‌ای نخواهد داشت.

۲۹. قرارداد API

تمام API ها باید از استاندارد REST پیروی کنند.

فرمت تبادل داده‌ها باید JSON باشد.

تمام پاسخ‌ها باید دارای Header زیر باشند.

Content-Type: application/json

۳۰. #ایجاد درخواست تسهیلات

Endpoint

http
POST /api/v1/loans

Request Body

json
{

```
  "customerId": "C-1001",
  "amount": 400000000,
  "phone": "09121234567",
  "loanType": "PERSONAL",
  "monthlyIncome": 50000000,
  "creditScore": 720,
  "hasGuarantor": false
}
```

##توضیح فیلدها

فیلد	الزامی	توضیح
customerId	بله	شناسه مشتری
amount	بله	مبلغ درخواست
phone	بله	شماره موبایل
loanType |بله PERSONAL	یا| BUSINESS
monthlyIncome	بله	درآمد ماهانه
creditScore	بله	امتیاز اعتباری
hasGuarantor	بله	وجود ضامن

Response

http
201 Created

json
```
{
   "loanId":"L-10001",
   "status":"SUBMITTED",
   "currentStage":"VALIDATION"
}
```

##خطاها

در صورتی که JSON ارسالی قابل Parse نباشد:

http
400 Bad Request

json
```
{
   "error":"INVALID_REQUEST"
}
```

نکته:

اعتبارسنجی مقادیر) مانند amount یا (phone داخل Workflow انجام می‌شود و نباید باعث خطای 400 شود.

۳۱. اجرای Workflow

Endpoint

http
POST /api/v1/loans/{loanId}/process

##توضیح

این API باید پردازش درخواست را از مرحله فعلی آغاز کند.

پردازش تا رسیدن به یکی از وضعیت‌های زیر ادامه پیدا می‌کند.

APPROVED

REJECTED

MANUAL_REVIEW

Response

http
200 OK

json
{
  "loanId":"L-10001",
  "status":"APPROVED",
  "currentStage":null
}

اگر وضعیت برابر MANUAL_REVIEW باشد

json
{
  "loanId":"L-10001",
  "status":"MANUAL_REVIEW",
  "currentStage":null
}

اگر شناسه وجود نداشته باشد

http
404 Not Found

```
json
{
  "error":"LOAN_NOT_FOUND"
}
```

۳۲.رفتار اجرای مجددProcess

اگر درخواست قبلاً APPROVED یا REJECTED یا MANUAL_REVIEW شده باشد، اجرای مجدد

POST /process نباید باعث اجرای دوباره Workflow شود. پاسخ باید وضعیت فعلی را برگرداند.

۳۳.دریافت اطلاعات درخواست

Endpoint

```
http
GET /api/v1/loans/{loanId}
```

Response

```
json
{
  "loanId":"L-10001",
  "customerId":"C-1001",
  "amount":400000000,
  "phone":"09121234567",
  "loanType":"PERSONAL",
  "monthlyIncome":50000000,
  "creditScore":720,
  "hasGuarantor":false,

  "status":"APPROVED",

  "currentStage":null,

  "createdAt":"2026-07-15T10:00:00Z",

  "updatedAt":"2026-07-15T10:00:03Z"
}
```

اگر شناسه وجود نداشته باشد

```
http
404

json
{
  "error":"LOAN_NOT_FOUND"
```

}

۳۴.دریافت تاریخچه پردازش

Endpoint

http
GET /api/v1/loans/{loanId}/history

Response

```
json
[
  {
    "stage":"VALIDATION",
    "result":"PASS",
    "timestamp":"2026-07-15T10:00:00Z",
    "reason":"SUCCESS"
  },
  {
    "stage":"FRAUD_CHECK",
    "result":"PASS",
    "timestamp":"2026-07-15T10:00:01Z",
    "reason":"SUCCESS"
  },
  {
    "stage":"CREDIT_CHECK",
    "result":"PASS",
    "timestamp":"2026-07-15T10:00:02Z",
    "reason":"SUCCESS"
  }
]
```

##ترتیب خروجی

تاریخچه باید بر اساس زمان اجرا مرتب شود.

هر Stage فقط یک بار ثبت می‌شود.

۳۵. Health Check

Endpoint

http
GET /health

Response

http
200 OK

json
```
{
   "status":"UP"
}
```

۳۶.فرمت زمان

تمامTimestamp ها باید مطابق استاندارد زیر باشند.

ISO-8601 UTC

نمونه

2026-07-15T10:00:00Z

۳۷.کدهای خطا

سیستم حداقل باید از کدهای زیر استفاده کند.

Code	توضیح
INVALID_REQUEST	JSON |نامعتبر
LOAN_NOT_FOUND	شناسه درخواست وجود ندارد
INVALID_AMOUNT	مبلغ نامعتبر
INVALID_PHONE	شماره موبایل نامعتبر
INVALID_CUSTOMER_ID	شناسه مشتری نامعتبر
INVALID_LOAN_TYPE	نوع وام نامعتبر
INVALID_CREDIT_SCORE	امتیاز اعتباری نامعتبر
INVALID_MONTHLY_INCOME	درآمد نامعتبر

۳۸.قراردادPersistence

پس از Restart برنامه، اطلاعات زیر باید بدون تغییر باقی بمانند.

-اطلاعات درخواست
-وضعیت فعلی
-مرحله فعلی

-تاریخچه مراحل

راهکار ذخیره‌سازی آزاد است.

۳۹.قراردادDocker

پروژه باید تنها با دستورات زیر قابل اجرا باشد.

```
bash
docker build -t bankflow .
```

سپس

```
bash
docker run -p 8080:8080 bankflow
```

سرویس باید حداکثر ظرف ۶۰ ثانیه آماده پاسخ‌گویی شود.

۴۰.ساختار فایل‌های تحویلی

حداقل فایل‌های زیر باید وجود داشته باشند.

```
project/ project/
                    |
                    ├── src/
                ├── Dockerfile
            ├── README.md
                ├── DESIGN.md
        ├── ENGINEERING_DECISIONS.md
                └── TESTING.md
```

۴۱. README.md

READMEباید شامل موارد زیر باشد.

-روشBuild
-روش اجرا
-روش اجرای تست‌ها
-توضیح کوتاه ساختار پروژه

-وابستگی‌های اصلی

۴۲. DESIGN.md

حداکثر سه صفحه.

این سند باید حداقل به پرسش‌های زیر پاسخ دهد.

۱- Workflow چگونه مدل شده است؟

۲-مرحله بعد چگونه انتخاب می‌شود؟

۳-برای اضافه شدن Stage جدید چه تغییری لازم است؟

۴-قوانین کسب‌وکار چگونه از فایل تنظیمات خوانده می‌شوند؟

۵-اطلاعات چگونه ذخیره می‌شوند؟

۶-اگر سیستم فردا بخواهد AML_CHECK اضافه کند چه تغییراتی لازم است؟

۷-سه تصمیم مهم طراحی خود را توضیح دهید.

۸-مهم‌ترینTrade-off های طراحی شما چه بوده است؟

---

۴۳. #محدودیت‌ها

استفاده از فناوری‌های زیر اختیاری است و امتیاز مستقیمی ندارد.

- Kafka
- RabbitMQ
- Redis
- Microservice
- Kubernetes
- CQRS
- Event Sourcing

امتیاز بر اساس کیفیت راهکار داده می‌شود، نه پیچیدگی فناوری‌های استفاده شده.

۴۴. #مستند تصمیم‌های مهندسی(ENGINEERING_DECISIONS.md)

علاوه بر مستند DESIGN.md، هر تیم باید فایل دیگری با نام زیر ارائه کند:

ENGINEERING_DECISIONS.md

هدف این مستند، آشنایی تیم داوری با نحوه تحلیل مسئله و تصمیم‌های مهندسی تیم است.

حداکثر حجم این فایل **دو صفحه** است.

##ساختار پیشنهادی

این مستند باید حداقل به پرسش‌های زیر پاسخ دهد.

۱. ###معماری انتخاب‌شده

معماری کلی سیستم چیست؟

چرا این معماری انتخاب شده است؟

در صورت وجود گزینه‌های دیگر، چرا انتخاب نشده‌اند؟

---

۲. ###مدل‌سازیWorkflow

Workflowچگونه مدل شده است؟

اگر بانک مرحله جدیدی اضافه کند، چه بخش‌هایی از سیستم تغییر خواهند کرد؟

---

۳. ###قوانین کسب‌وکار

قوانین چگونه از فایل تنظیمات خوانده می‌شوند؟

چرا این روش انتخاب شده است؟

---

۴. ###مدیریت وضعیت(State Management)

وضعیت فعلی درخواست چگونه مدیریت می‌شود؟

چگونه از اجرای مجدد مراحل جلوگیری شده است؟

---

۵. ###ماندگاری اطلاعات

چه مکانیزمی برای ذخیره اطلاعات انتخاب شده است؟

چرا این روش انتخاب شده است؟

---

۶. ###مهم‌ترینTrade-off ها

سه تصمیم مهم طراحی خود را توضیح دهید.

برای هر تصمیم موارد زیر را بیان کنید.

-گزینه‌های موجود
-گزینه انتخاب‌شده
-دلیل انتخاب
-مزایا
-محدودیت‌ها

۷. ###محدودیت‌های فعلی

اگر زمان بیشتر داشتید، چه بخش‌هایی از سیستم را بهبود می‌دادید؟

۸. ###برنامه توسعه آینده

فرض کنید بانک درخواست‌های زیر را مطرح کند.

-اضافه شدن مرحلهAML
-اضافه شدن مرحلهLegal Check
-اضافه شدن نوع جدیدی از تسهیلات
-تغییر Workflow بانک‌ها

توضیح دهید معماری فعلی چگونه از این تغییرات پشتیبانی می‌کند.

#نکات

این مستند نباید صرفاً شامل نمودارهای معماری باشد.

هدف این فایل، نمایش نحوه تفکر مهندسی تیم است.

کیفیت تحلیل و استدلال از حجم مستند مهم‌تر است.

۴۵. مستنداستراتژی و پوشش تست‌ها(`TESTING.md`)

در این فایل تیم باید توضیح دهد:

- چه تست‌هایی نوشته است؟
- چه بخش‌هایی را Unit Test کرده است؟
- چه بخش‌هایی Integration Test شده‌اند؟
- چه بخش‌هایی را به دلیل محدودیت زمان تست نکرده است؟

۴۶. نکات پایانی

این چالش یک مسئله طراحی نرم‌افزار است.

هیچ پاسخ واحد یا معماری از پیش تعیین‌شده‌ای وجود ندارد.

از شرکت‌کنندگان انتظار می‌رود تصمیم‌های مهندسی خود را مستند کرده و در جلسه داوری از آن‌ها دفاع کنند.

.۴۶ #ساختار پروژه

ساختار پروژه آزاد است و شرکت‌کنندگان می‌توانند بر اساس زبان و معماری انتخابی خود، ساختار مناسبی طراحی کنند.

حداقل فایل‌های زیر باید در پروژه وجود داشته باشند.

```
project/
│
├── src/
├── Dockerfile
├── README.md
├── DESIGN.md
├── ENGINEERING_DECISIONS.md
└── TESTING.md
```

در صورت نیاز می‌توانید فایل‌های دیگری نیز به پروژه اضافه کنید.

۴۷. Docker

پروژه باید با استفاده از Docker قابل اجرا باشد.

حداقل دستورات زیر باید بدون نیاز به تغییر در پروژه اجرا شوند.

ساختImage

```bash
docker build -t bankflow .
```

اجرای پروژه

```bash
docker run -p 8080:8080 bankflow
```

پروژه باید حداکثر ظرف ۶۰ ثانیه آماده پاسخ‌گویی شود.

---

۴۸. #فایلREADME.md

READMEمهم‌ترین راهنمای داور برای اجرای پروژه است.

حداقل اطلاعات زیر باید در آن وجود داشته باشد.

##معرفی پروژه

توضیح کوتاهی درباره راهکار ارائه شده.

##پیش‌نیازها

در صورت وجود وابستگی خاص توضیح داده شود.

Build

روش Build پروژه.

Run

روش اجرای پروژه.

##اجرای تست‌ها

توضیح نحوه اجرای تست‌ها.

##ساختار پروژه

معرفی پوشه‌های اصلی پروژه.

##فناوری‌های استفاده شده

فهرست زبان Database ،Framework و سایر فناوری‌های اصلی.

---

۴۹. #فایلDESIGN.md

حداکثر سه صفحه.

این فایل باید شامل موارد زیر باشد.

-معرفی معماری
-اجزای اصلی سیستم
-نحوه اجرایWorkflow
-نحوه توسعهWorkflow
-نحوه ذخیره اطلاعات
-نحوه مدیریت وضعیت‌ها
-نحوه جلوگیری از پردازش تکراری
-مهم‌ترین تصمیم‌های طراحی

---

۵۰. #فایلENGINEERING_DECISIONS.md

حداکثر دو صفحه.

این فایل باید پاسخگوی پرسش‌های زیر باشد.

-چرا این معماری انتخاب شده است؟
-چه گزینه‌های دیگری وجود داشت؟
-مهم‌ترینTrade-off های طراحی چیست؟
-اگر یک هفته زمان بیشتر داشتید چه تغییراتی اعمال می‌کردید؟
-بزرگ‌ترین محدودیت راهکار فعلی چیست؟

---

۵۱. #فایلTESTING.md

حداکثر یک صفحه.

در این فایل توضیح دهید.

-چه تست‌هایی نوشته شده است؟
-چه بخش‌هایی Unit Test شده‌اند؟
-چه بخش‌هایی Integration Test شده‌اند؟
-چه بخش‌هایی به دلیل محدودیت زمان تست نشده‌اند؟

---

۵۲. #محدودیت‌های فنی

شرکت‌کنندگان در انتخاب فناوری آزاد هستند.

استفاده از موارد زیر اختیاری است.

- Java
- Kotlin
- C#
- Go
- Rust
- Python
- Node.js
- PHP
- C++

همچنین استفاده از Framework، ORM، Database و سایر ابزارها آزاد است.

۵۳.موارد خارج از دامنه

پیاده‌سازی موارد زیر الزامی نیست.

-رابط کاربری(Frontend)
- Authentication
- Authorization
- Message Broker
- Kafka
- RabbitMQ
- Redis
- Kubernetes
- Cloud Deployment
- CI/CD
- Distributed Transaction
- Notification
- Core Banking Integration
-سیستم واقعیManual Review

استفاده از این فناوری‌ها امتیاز مستقیمی ایجاد نخواهد کرد.

۵۴.محدودیت‌های اجرایی

-پروژه باید تنها با استفاده از Docker اجرا شود.
-سرویس باید روی پورت 8080 در دسترس باشد.
-تمامAPI ها باید JSON باشند.
-زمان آماده شدن سرویس نباید بیش از ۶۰ ثانیه باشد.
Timestamp -ها باید مطابق استاندارد ISO-8601 UTC باشند.

.۵۵آنچه در این چالش اهمیت دارد

این چالش یک مسئله طراحی نرم‌افزار است.

موارد زیر بیشترین اهمیت را دارند.

-درستی عملکرد سیستم
-طراحی مناسب
-توسعه‌پذیری
-خوانایی کد
-قابلیت نگهداری
-قابلیت تست
-سادگی طراحی
-مستندسازی مناسب

پیچیدگی بیشتر، امتیاز بیشتر به همراه نخواهد داشت.

.۵۶ #پرسش‌های متداول(FAQ)

##آیا استفاده از هوش مصنوعی مجاز است؟

بله.

استفاده از ابزارهای هوش مصنوعی مجاز است.

با این حال، تیم باید بتواند در جلسه داوری از طراحی و پیاده‌سازی خود دفاع کند.

##آیا استفاده از Microservice امتیاز دارد؟

خیر.

امتیاز بر اساس کیفیت طراحی داده می‌شود، نه نوع معماری.

##آیا استفاده از Kafka یا RabbitMQ امتیاز دارد؟

خیر.

در صورت استفاده، تیم باید دلیل فنی این انتخاب را توضیح دهد.

##آیا استفاده از Framework خاص امتیاز دارد؟

خیر.

انتخاب فناوری کاملاً آزاد است.

##آیا استفاده از SQLite مجاز است؟

بله.

هر مکانیزم ذخیره‌سازی که نیازمندی‌های مسئله را برآورده کند، قابل قبول است.

##آیا استفاده از فایل به جای پایگاه داده مجاز است؟

بله.

در صورتی که تمام نیازمندی‌های Persistence رعایت شود.

##آیا ظاهر پروژه در امتیاز تأثیر دارد؟

خیر.

رابط کاربری بخشی از این چالش نیست.

##آیا استفاده از کتابخانه‌های متن‌باز مجاز است؟

بله.

استفاده از کتابخانه‌های متن‌باز مجاز است.

##آیا لازم است همه قابلیت‌ها پیاده‌سازی شوند؟

توصیه می‌شود ابتدا تمام نیازمندی‌های اصلی به‌صورت صحیح پیاده‌سازی شوند.

پیاده‌سازی ناقص قابلیت‌های اصلی با افزودن قابلیت‌های جانبی جبران نخواهد شد.

هدف این چالش، طراحی سامانه‌ای است که علاوه بر عملکرد صحیح، از نظر کیفیت مهندسی نیز قابل دفاع باشد.

راهکار ارائه شده باید نشان دهد تیم توانسته است:

-مسئله را به‌درستی تحلیل کند.
-معماری مناسبی انتخاب کند.
-مسئولیت‌ها را به‌درستی تفکیک کند.
-کدی خوانا و قابل توسعه تولید کند.
-تصمیم‌های مهندسی خود را مستند و قابل دفاع ارائه دهد.

موفق باشید.

اقلام تحویلی باید به این شکل باشد:

هدف	فایل
پیاده‌سازی	Source Code
اجرای پروژه	Dockerfile
راهنمای اجرا	README.md
معرفی معماری	DESIGN.md
تحلیل و تصمیم‌های مهندسی	ENGINEERING_DECISIONS.md
استراتژی و پوشش تست‌ها	TESTING.md