

This document describes all the details pertaining to the Analytics Reporter project (Take home assignment by Agoda to **Bhaswant Inguva**).

Set Up

- 1. Unzip the folder to the see the following contents.
 - a. AgodaSubmission.zip
 - b. AgodaSubmission_Sources.zip
 - c. jacoco-ui.zip
 - d. ReadMe.pdf
- 2. Extract AgodaSubmission.zip to see analytics-reporter.jar
- 3. Run the jar using the command:
`java -jar analytics-reporter.jar`
- 4. You should see the application up and running with the following logs:

```
Wed Jul 28 16:12:59 180808@B:~/Desktop/Prep/Agoda/analytics-reporter/target$ java -jar analytics-reporter.jar
:: Spring Boot :: (v2.3.2)

2021-07-28 16:13:04.969 INFO 46721 --- [main] c.a.i.a.AnalyticsReporterApplication : Starting AnalyticsReporterApplication v0.8.1-SNAPSHOT using Java 1.8.0_271 on m-c82d2ap4nd6n with PID 46721 (/Users/180808@B/Desktop/Prep/Agoda/analytics-reporter/target/analytics-reporter.jar started by 180808@B in /Users/180808@B/Desktop/Prep/Agoda/analytics-reporter/target)
2021-07-28 16:13:04.971 INFO 46721 --- [main] c.a.i.a.AnalyticsReporterApplication : The following profiles are active: dev
2021-07-28 16:13:05.192 INFO 46721 --- [main] s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2021-07-28 16:13:05.847 INFO 46721 --- [main] s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 44 ms. Found 1 JPA repository interfaces.
2021-07-28 16:13:06.411 INFO 46721 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2021-07-28 16:13:06.422 INFO 46721 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2021-07-28 16:13:06.422 INFO 46721 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.48]
2021-07-28 16:13:06.684 INFO 46721 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2021-07-28 16:13:06.684 INFO 46721 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 1651 ms
2021-07-28 16:13:06.713 INFO 46721 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2021-07-28 16:13:06.994 INFO 46721 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2021-07-28 16:13:07.001 INFO 46721 --- [main] o.s.b.a.h2.H2ConsoleAutoConfiguration : H2 console available at '/h2-console'. Database available at 'jdbc:h2:mem:testdb'
2021-07-28 16:13:07.191 INFO 46721 --- [main] o.hibernate.jpa.internal.util.LogHelper : HH000204: Processing PersistenceUnitInfo [name: default]
2021-07-28 16:13:07.264 INFO 46721 --- [main] org.hibernate.Version : HH0000412: Hibernate ORM core version 5.4.32.Final
2021-07-28 16:13:07.402 INFO 46721 --- [main] o.hibernate.annotations.common.Version : HCANN000001: Hibernate Commons Annotations (5.1.2.Final)
2021-07-28 16:13:07.517 INFO 46721 --- [main] org.hibernate.dialect.Dialect : HH0000400: Using dialect: org.hibernate.dialect.H2Dialect
2021-07-28 16:13:08.068 INFO 46721 --- [main] o.h.e.t.j.p.a.JpaPlatformInitiator : HH0000499: Using JpaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
2021-07-28 16:13:08.088 INFO 46721 --- [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2021-07-28 16:13:09.136 INFO 46721 --- [main] c.a.i.a.service.impl.BookingServiceImpl : Successfully imported seed data
2021-07-28 16:13:09.459 WARN 46721 --- [main] JpaBaseConfigurationsJpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may be performed during view rendering. Explicitly configure spring.jpa.open-in-view to disable this warning
2021-07-28 16:13:10.638 INFO 46721 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2021-07-28 16:13:10.846 INFO 46721 --- [main] d.s.w.p.DocumentationPluginsBootstrapper : Context refreshed
2021-07-28 16:13:10.865 INFO 46721 --- [main] d.s.w.p.DocumentationPluginsBootstrapper : Found 1 custom documentation plugin(s)
2021-07-28 16:13:10.975 INFO 46721 --- [main] s.d.s.w.s.ApiListingReferenceScanner : Scanning for api listing references
2021-07-28 16:13:10.129 INFO 46721 --- [main] d.s.w.r.o.CachingOperationNameGenerator : Generating unique operation named: getSummaryUsingGET_1
2021-07-28 16:13:10.242 INFO 46721 --- [main] c.a.i.a.AnalyticsReporterApplication : Started AnalyticsReporterApplication in 5.663 seconds (JVM running for 6.88)
```

API Endpoints & Use cases

- 1. Swagger UI is enabled for this project to help clarify and test the APIs. Navigate to `/swagger-ui.html` (Sample: <http://localhost:8080/swagger-ui.html>) . You should see a UI in the following format



Api Documentation

Api Documentation

Apache 2.0

basic-error-controller : Basic Error Controller	Show/Hide	List Operations	Expand Operations
booking-controller : Booking Controller	Show/Hide	List Operations	Expand Operations
customer-controller : Customer Controller	Show/Hide	List Operations	Expand Operations
hotel-controller : Hotel Controller	Show/Hide	List Operations	Expand Operations

- 2. API_1:
 - a. API: POST: `/v1/booking`
 - b. Description: API to create bookings into the database. Supported formats Json, CSV. If the input type is CSV, first line is expected to be headers
 - c. Headers: Possible values:
 - i. format: json
 - ii. format: CSV

d. Payload sample:

```
{
  "bookings": [{
    "hotel_id": "1234",
    "booking_id": "1000000",
    "customer_id": "4e9a8620-7791-4125--7229ff441be0",
    "selling_price_local_currency": 93,
    "currency": "INR",
    "to_usd_exchange_rate": 0.76
  }, {
    "hotel_id": "1234",
    "booking_id": "1000001",
    "customer_id": "7ac534a5-ec17-48a4-9149-cc260ee3a653",
    "selling_price_local_currency": 597,
    "currency": "SGB",
    "to_usd_exchange_rate": 0.03
  }]
}
```

3. API_2:

- a. API: GET: /v1/hotel?hotelId=<Id>&exchangeRate=<ExchangeRate>
- b. Description: This API accepts hotel Id as a request parameter which is used to fetch all bookings of the hotel across the booking data and aggregates the revenue it generated (in USD) through these bookings. There is also an optional exchange rate request parameter, which if provided will be used to convert the revenue to USD.
- c. Sample Response:

```
{
  "hotelId": "1000134",
  "numberOfBookings": 25,
  "totalPriceUsd": 9654.069999999998
}
```

4. API_3:

- a. API: GET: /v1/customer
- b. Description: Given one or more customer Ids, fetches all their bookings and their expenditure on these bookings (in USD, using the conversion rate of the Booking Data)
- c. Sample Request:

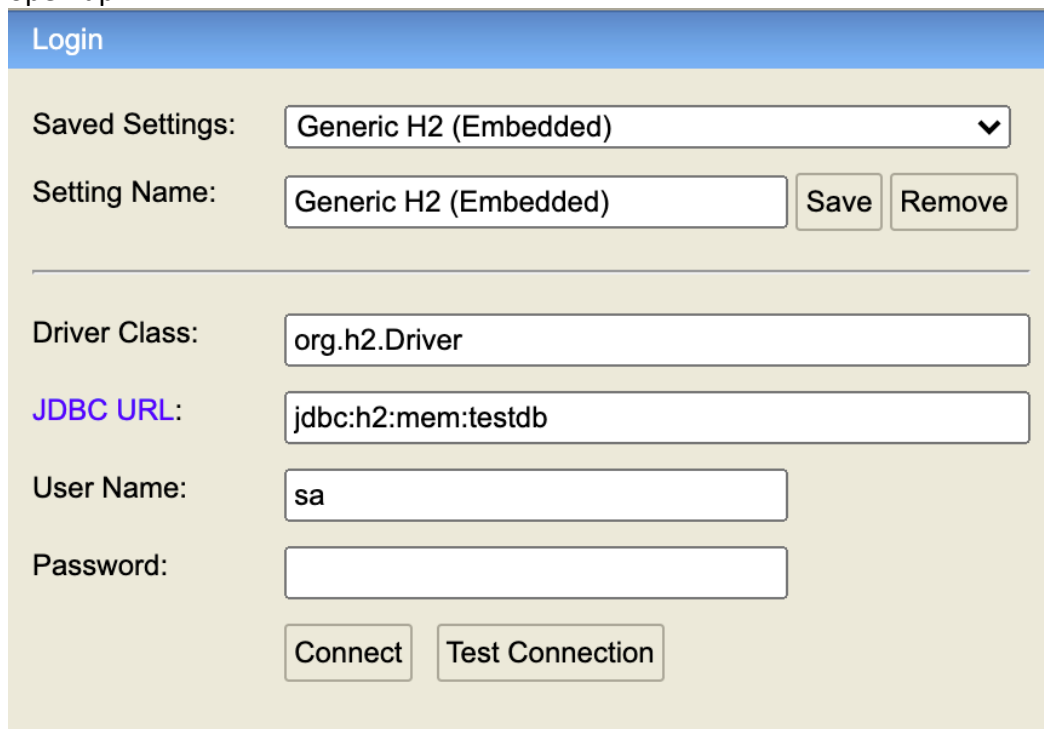
```
{
  "customer_ids": [
    "00c6cf82-d248-4b3f-86b2-8bd73ca856ee",
    "9e80a7c0-7ec4-4f6a-a799-"
  ]
}
```

d. Sample Response:

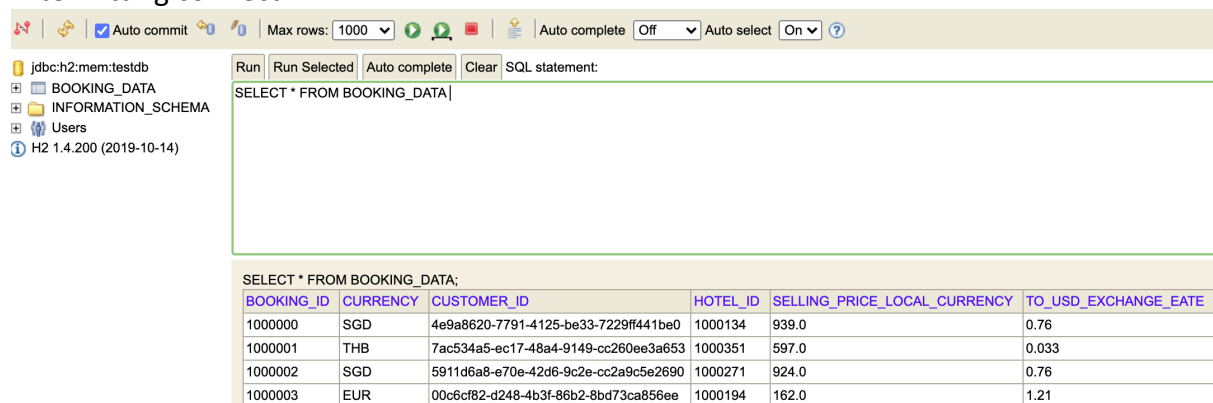
```
{
  {
    "customerId": "00c6cf82-d248-4b3f-86b2-8bd73ca856ee",
    "numberOfBookings": 9,
    "totalPriceUsd": 4136.138
  },
  {
    "customerId": "9e80a7c0-7ec4-4f6a-a799-",
    "numberOfBookings": 0,
    "totalPriceUsd": 0.0
  }
}
```

Assumptions

1. Used H2 for database capabilities. To access h2 client, navigate to [/h2-console](#) (Sample: <http://localhost:8080/h2-console>) where you can query h2 in SQL format. Following UI should open up



After hitting connect:



BOOKING_ID	CURRENCY	CUSTOMER_ID	HOTEL_ID	SELLING_PRICE_LOCAL_CURRENCY	TO_USD_EXCHANGE_EATE
1000000	SGD	4e9a8620-7791-4125-be33-7229ff441be0	1000134	939.0	0.76
1000001	THB	7ac534a5-ec17-48a4-9149-cc260ee3a653	1000351	597.0	0.033
1000002	SGD	5911d6a8-e70e-42d6-9c2e-cc2a9c5e2690	1000271	924.0	0.76
1000003	EUR	00c6cf82-d248-4b3f-86b2-8bd73ca856ee	1000194	162.0	1.21

2. While the API_3 was asked to be a POST request, since the requirement is to fetch the data and not to create it, converted it to a GET request with payload.
3. In API_2 if hotel asked for is not part of the BookingsData, returning a default object. Since, a hotel can exist and no bookings available for it and we don't have handle to hotel DB as of now, returning empty object

Sample:











```
{
  "hotelId": "2",
  "numberOfBookings": 0,
  "totalPriceUsd": 0
}
```

4. In API_2 if exchange rate is provided, using the same rate for all the bookings irrespective of the currency used. If not provided, using the corresponding row entry's exchange data.

Test Cases

1. Test cases comprising unit tests and integration tests, both positive and negative account for 90% test coverage.
 - a. To view this full report, unzip jacoco-ui.zip → jacoco/index.html
 - b. This can also be generated by building the sources file in the directory where pom.xml is available.

analytics-reporter

Element	Missed Instructions	Cov.	Missed Branches	Cov.
 com.agoda.interview.analyticsreporter.helper		90%		76%
 com.agoda.interview.analyticsreporter.service.impl		93%		100%
 com.agoda.interview.analyticsreporter		63%		n/a
 com.agoda.interview.analyticsreporter.exception		66%		n/a
Total	67 of 727	90%	3 of 21	85%

2. Full test resources and classes can be accessed at AgodaSubmission_Sources.zip → src/test