Project Deliverable 2
Software Quality
Group 06

Okikioluwa Ojo (100790236)
Abdullah Hanoosh (100749026)
Nancy Emanuel (100657804)
Mohammad Daiyan(100825241)

Due Date: Apr 2, 2024

GitHub Link: https://github.com/NancyEmanuel/SOFE-3980U-Project-Deliverable-1/tree/main

Ticket Booking Application

In your project, you will use the tools of CI/CD from the milestones and Test-driven development (TDD) to design a flight ticket booking application for an airline company. Using additional libraries or online resources in implementing the application should be minimal.

Following are some of the application requirements:

1. The application has a web interface.
2. The application allows users to book direct or multi-stop flights.
3. The application allows users to book a one-way or round trip.
4. The application contains only a list of weekly direct flights.
5. The application reports the total flight time.
6. A 24-hour format is used in the application.
7. The ticket may use a 12-hour or 24-hour format per user preferences.
8. The application does not produce any cyclic trips from the same airport.

The first deliverables include:

1. all the functional requirements for the application and any assumptions such as
    a. You can book a ticket
    b. You can pay with Money, Credit Cards, etc…
2. the design of project modules (e.g., classes) and the methods in each module.
3. the design of unit tests for each method.
4. the design of integration tests for modules.
5. a GitHub repo with the initial implementation of the unit and integration tests.
6. the role of each group member in the deliverables (detailed tasks and percentage)

Functional Requirements

1. **User Interface**: A web interface that's easy to use for viewing weekly non cyclic flights and booking flights
2. **Flight Booking**:
    - Support for booking flights by booking purchasing flight tickets for 1 or multiple passengers for a direct, multistop, roundtrip, or one way flight
    - Option of Booking Tickets using the 12/24 hour formats
3. **Flight Scheduling**: A list of upcoming flights along with their destinations, departure dates, arrive dates, and time format in 12 or 24 according to user preference
4. **Flight Paths Planning**: Users should be able to use the application to plan out which flights they'd like to take by navigating on the booking interface and viewing the different airport options and departure and arrive dates

Assumptions

- Flight ticket prices are relatively static and do not change based on demand or time of booking so they are not included
- That the user can select the flights they require from the list of available flights

Changes to Initial Design

- We streamlined our design to include a user friendly UI, which allows users to select a departure and arrival airport from our list of airports, then use a dropdown date picker to select their departure and arrival date
- We also added a 12-14 hour option so the user can select how they would like their ticket information to be displayed
- We added a page which displays all available flights and then gives users the option to select the flight they are interested in and click "book now" to book the flight and generate their ticket

List of Available Flights

Note: refer to these when testing the system

1. Toronto pearson and LA, departure date April 1, 2024, arrive date april 2, 2024
2. Toronto Pearson to Heathrow, departure date april 3, 2024, arrive date april 4, 2024
3. LA to Dubai, departure date april 5, 2024, arrive date april 6, 2024

Design of Project Modules

The application can be divided into several modules for ease of development and maintenance:

1. **User Auth Module**: Handles user interface, allows users to input the departure and arrival airports, departure and arrival times
2. **Event Management Module for ticket**: Manages the ticket view and confirmation on the Thymeleaf TicketConfirmation UI, displays ticket information such as the departure and arrival airport, flight time, and dates
3. **UI Module**: Displays the web interface for user interaction.
4. **Utility Module**: Offers utility functions like time format conversion and math calculations for tickets, etc…., present in bookflightsinterfacecontroller

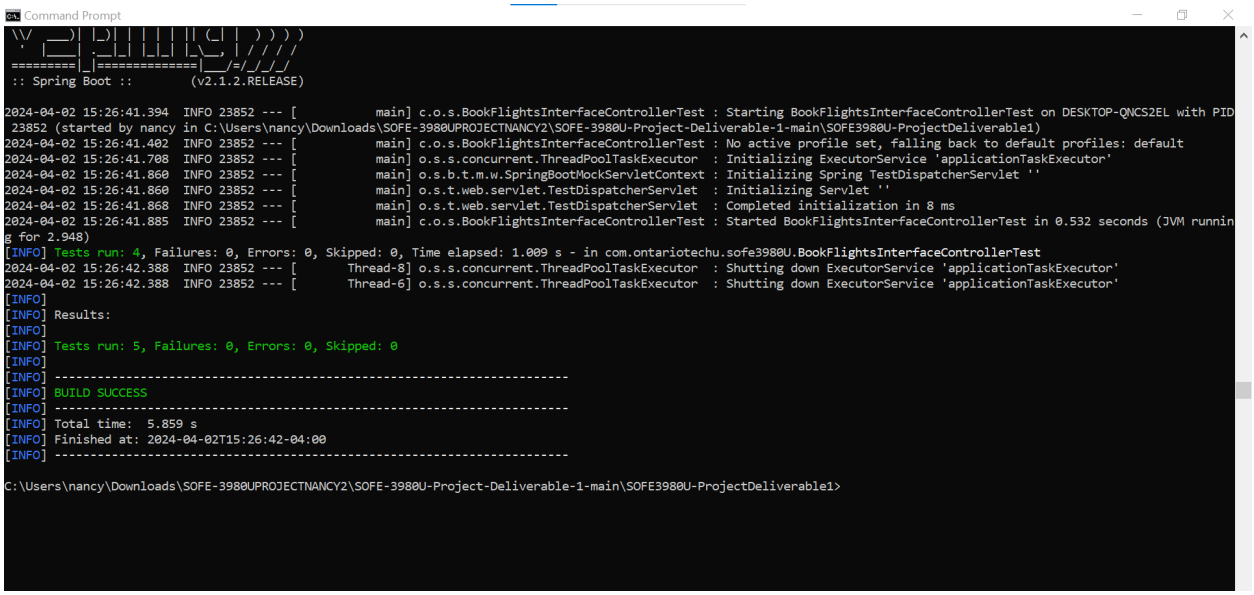
Design of Unit Tests (BookFlightsInterfaceControllerTest)

Each method within the modules should have corresponding unit tests to ensure functionality and detect regressions early. For example:

- Verify view tests :Testing of the different views including the booking interface and available flights
- whenFullNameProvided_thenCorrectModelAndView: test verifiys that when a request is sent to the rooturl with the FullName parameter, the controller processes the request and returns our BookingInterface with the fullname parameter set to "Jane Doe"
- whenFullNameNotProvided_thenDefaultModelAndView: test checks when request made to the url without the FullName parameter filled out .

- Getbookingview: When GET request made to the rooturl, the interfacecontroller returns BookingInterface view, with status OK (200)

Unit Tests Results

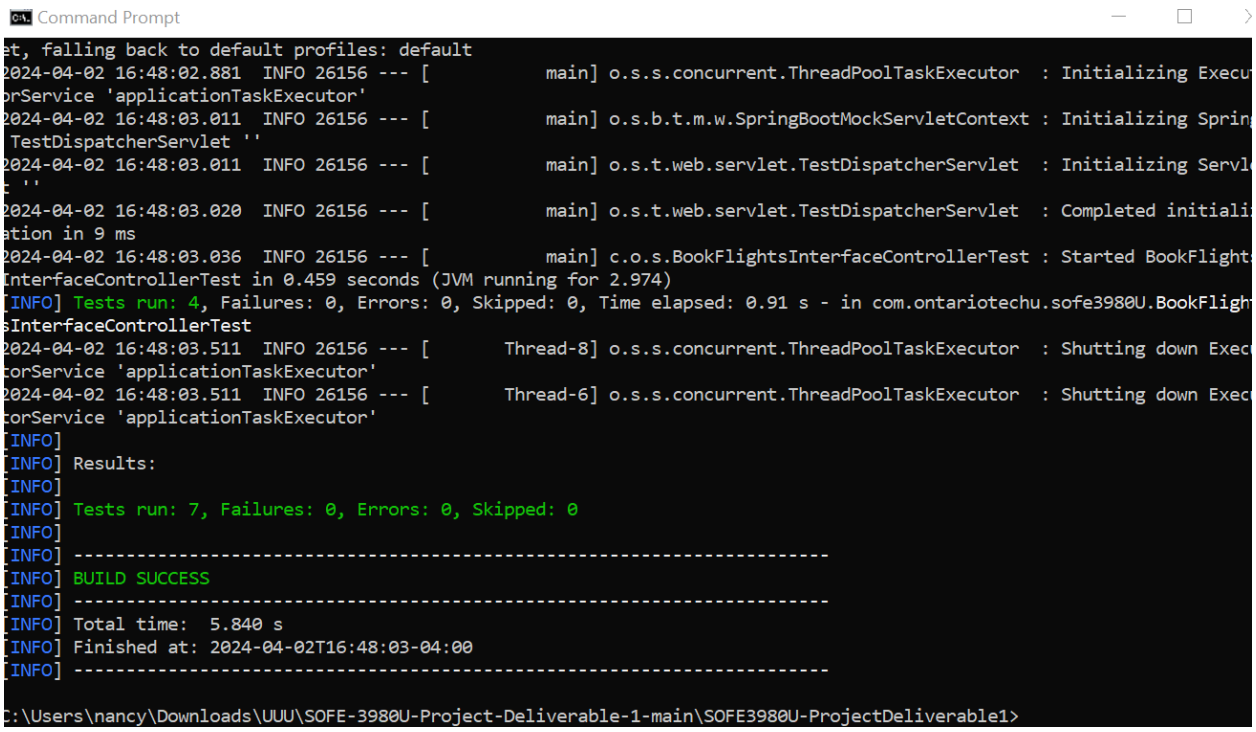


Design of Integration Tests(BookFlightsAPIControllerTest)

Integration tests should focus on the interaction between modules:

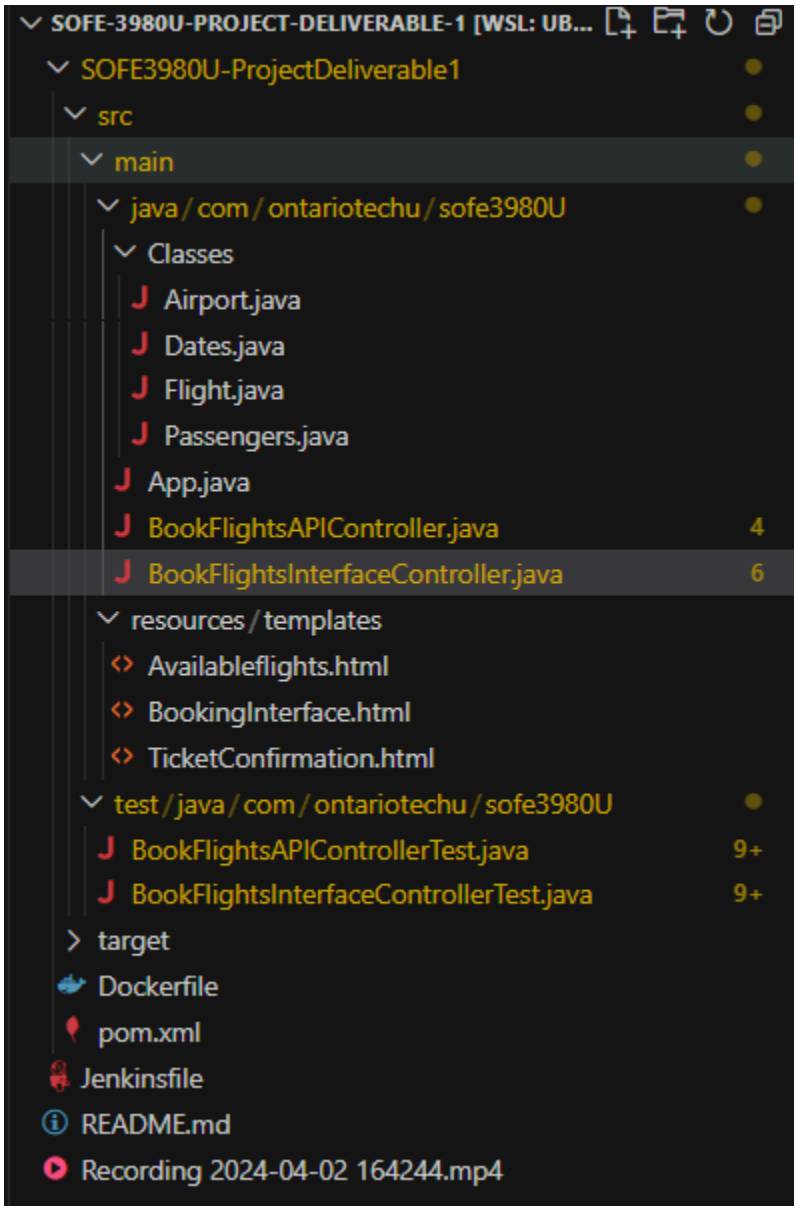
- Bookdirectapi: Tests for when booking direct flight with valid airport names
- bookDirectblankairportthenOk: Test for when booking direct flight with blank "FromAirport", response should be OK
- bookBlankDestinationAirportOk: Test for when booking direct flight with blank DestinationAirport, response should be OK

Integration Test Results


Command Prompt
et, falling back to default profiles: default
2024-04-02 16:48:02.881 INFO 26156 --- [main] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing Execu
orService 'applicationTaskExecutor'
2024-04-02 16:48:03.011 INFO 26156 --- [main] o.s.b.t.m.w.SpringBootMockServletContext : Initializing Sprin
TestDispatcherServlet ''
2024-04-02 16:48:03.011 INFO 26156 --- [main] o.s.t.web.servlet.TestDispatcherServlet : Initializing Servl
''
2024-04-02 16:48:03.020 INFO 26156 --- [main] o.s.t.web.servlet.TestDispatcherServlet : Completed initiali
ation in 9 ms
2024-04-02 16:48:03.036 INFO 26156 --- [main] c.o.s.BookFlightsInterfaceControllerTest : Started BookFlight
InterfaceControllerTest in 0.459 seconds (JVM running for 2.974)
[INFO] Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.91 s - in com.ontariotechu.sofe3980U.BookFligh
sInterfaceControllerTest
2024-04-02 16:48:03.511 INFO 26156 --- [Thread-8] o.s.s.concurrent.ThreadPoolTaskExecutor : Shutting down Exec
torService 'applicationTaskExecutor'
2024-04-02 16:48:03.511 INFO 26156 --- [Thread-6] o.s.s.concurrent.ThreadPoolTaskExecutor : Shutting down Exec
torService 'applicationTaskExecutor'
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 7, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] --
[INFO] BUILD SUCCESS
[INFO] --
[INFO] Total time: 5.840 s
[INFO] Finished at: 2024-04-02T16:48:03-04:00
[INFO] --
C:\Users\nancy\Downloads\UUU\SOFE-3980U-Project-Deliverable-1-main\SOFE3980U-ProjectDeliverable1>

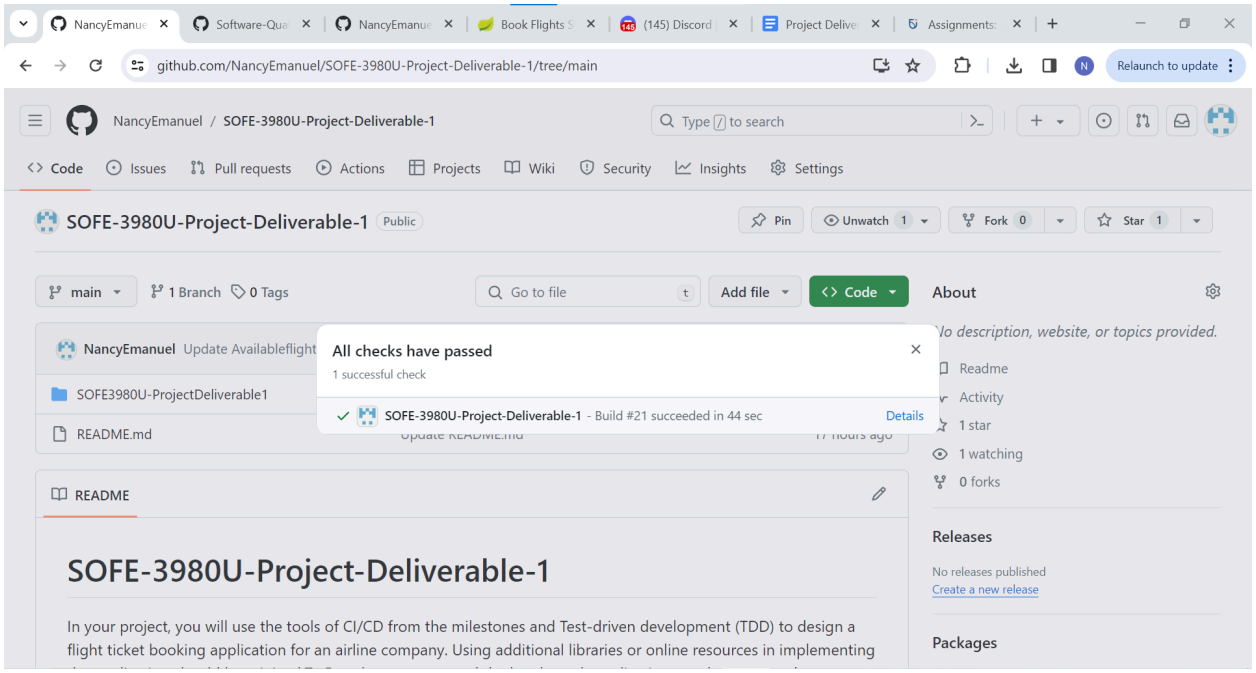
GitHub Repository Initial Setup

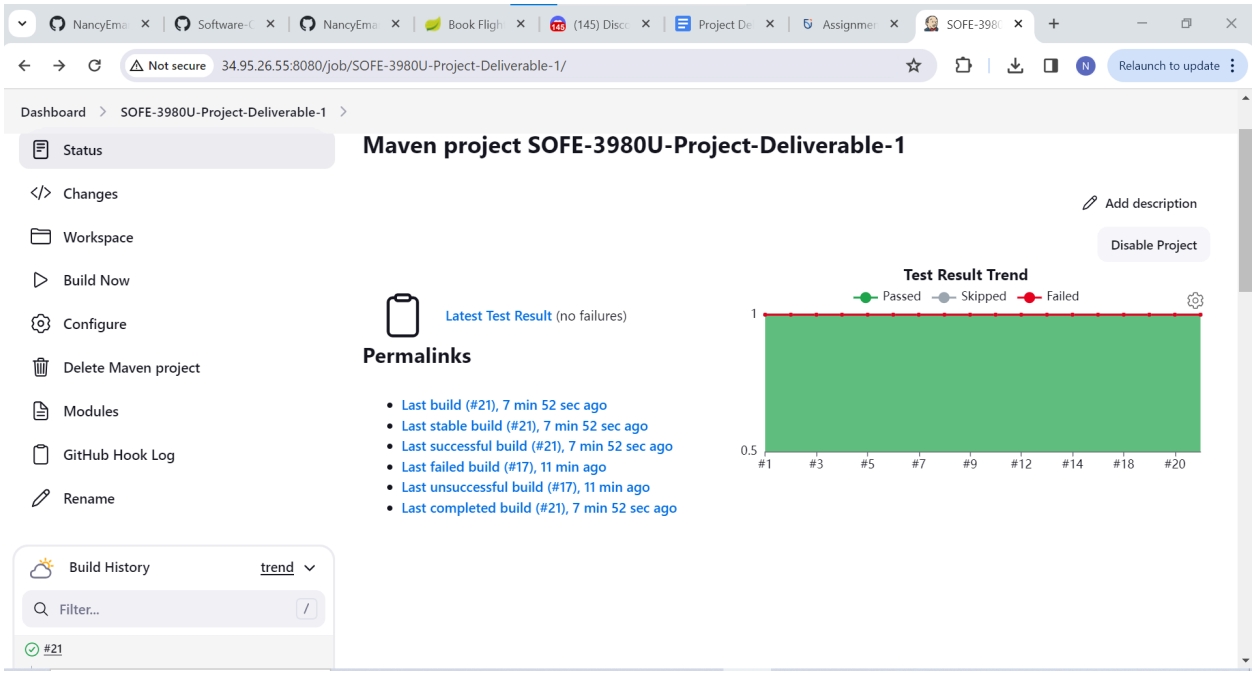
We structured the repo following the Java convention, with a src, and target folder with the main, resources and test folders inside the src folder.



We're using Spring Boot to setup the application, our goal is to have APIs, Controllers, Models, and the TicketBooking User Interface via HTML, and CSS.

CI PIPELINE SETUP


github.com/NancyEmanuel/SOFE-3980U-Project-Deliverable-1/tree/main
NancyEmanuel / SOFE-3980U-Project-Deliverable-1
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
SOFE-3980U-Project-Deliverable-1
Public
Pin
Unwatch 1
Fork 0
Star 1
main
1 Branch
0 Tags
Go to file
Add file
Code
About
No description, website, or topics provided.
Readme
Activity
1 star
1 watching
0 forks
NancyEmanuel Update Availableflight
SOFE3980U-ProjectDeliverable1
README.md
All checks have passed
1 successful check
SOFE-3980U-Project-Deliverable-1 - Build #21 succeeded in 44 sec
Details
README
SOFE-3980U-Project-Deliverable-1
In your project, you will use the tools of CI/CD from the milestones and Test-driven development (TDD) to design a flight ticket booking application for an airline company. Using additional libraries or online resources in implementing
Releases
No releases published
Create a new release
Packages


Not secure
34.95.26.55:8080/job/SOFE-3980U-Project-Deliverable-1/
Dashboard
SOFE-3980U-Project-Deliverable-1
Status
Changes
Workspace
Build Now
Configure
Delete Maven project
Modules
GitHub Hook Log
Rename
Maven project SOFE-3980U-Project-Deliverable-1
Add description
Disable Project
Latest Test Result (no failures)
Permalinks
Last build (#21), 7 min 52 sec ago
Last stable build (#21), 7 min 52 sec ago
Last successful build (#21), 7 min 52 sec ago
Last failed build (#17), 11 min ago
Last unsuccessful build (#17), 11 min ago
Last completed build (#21), 7 min 52 sec ago
Test Result Trend
Passed
Skipped
Failed
1
0.5
#1
#3
#5
#7
#9
#12
#14
#18
#20
Build History
trend
Filter...
#21

Continuous Deployment

Unfortunately we weren't able to get the Continuous Deployment working, but we were successful in getting Continuous Integration working.

Demo Video

I've added a demo video to the GitHub Repository.

Role of each group member in the **deliverable 2** (detailed tasks and percentage)

1. **Web interface: Thymeleaf html templates 20% :** Nancy Emanuel , Abdullah Hanoosh
2. **Backend: Booking Interface Controller module, Booking API Controller module 20%:** Nancy Emanuel , Abdullah Hanoosh, Okiki Ojo
3. **Updated project modules/classes (Flights, Airport, etc) 10%:** Nancy Emanuel , Abdullah Hanoosh
4. **Updated Unit Testing 10%:** Abdullah Hanoosh, Nancy Emanuel, Okiki Ojo
5. **Updated Integration Testing 10%:**
6. **CI pipeline 10%:** Nancy Emanuel
7. **CD pipeline & Demo Video 10%:** Okiki Ojo
