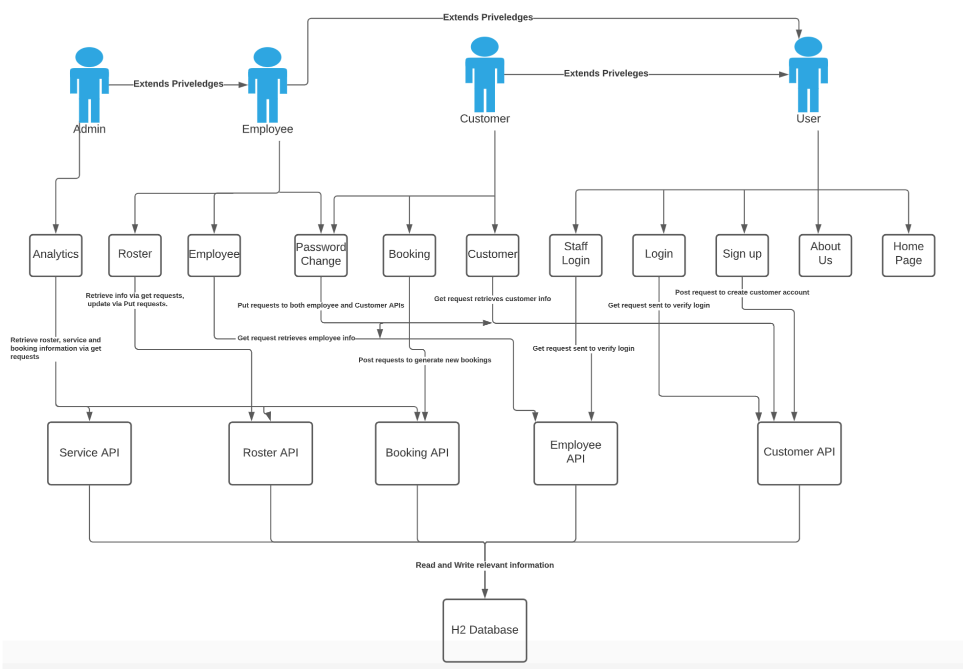
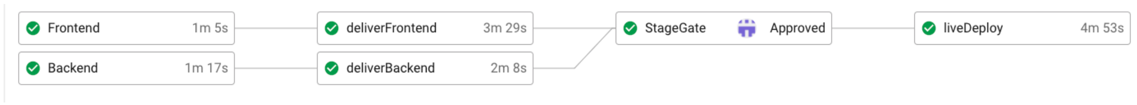
- a "Vision" statement, i.e., a description of the why the product is "valuable" (about 1/2 page)
    - **Vision statement**
    - Our booking service has been created with ease of use and business functionality at the forefront of our thinking throughout production. Our custom-made application has been designed to streamline the booking process, and is capable of maintaining the general business operations for a wide variety of services, giving business owners the power to generate new staff accounts, alter their weekly operating hours and get an overview on their current booking and staffing loads. In addition to this, employees have the capability of requesting rostering changes, cancelling bookings, and updating their information. Users are quickly and effectively able to create accounts, meaning new users are only ever a few clicks away from becoming new customers, after which they can make bookings and customize their account info. With this in mind, our AGME booking service remains consistent with business rules, as one of the key features of our application is its ability to remain consistent for both employees and customers, preventing potential pitfalls such as the creation of duplicate bookings, duplicate accounts, accounts with invalid information or bookings inconsistent with trading hours. From this, our first product iteration has sufficient functionality to meet most business needs, but more importantly during our build process we put in place our continuous deployment pipeline, meaning we have the means to continuously provide software updates quickly and easily. This means after hearing feedback from all users, customers, employees, and service owners alike, we can tailor our development to suit their needs, then automatically re-deploy our application quickly whilst minimizing the risk of deployment issues. Overall, our application is quite intuitive to use, has been tested extensively and has been designed and created with the agile philosophy in mind, that is a build process that always emphasizes usability and the creation of working software.
- a figure showing the final basic system architecture / design --- this does not need to be a UML diagram: a box-and-lines sketch architecture is fine; focus on the 3 tiers (front, backend, datastore) and what components are in each


Extends Priveledges
Admin
Extends Priveledges
Employee
Customer
Extends Priveleges
User
Analytics
Roster
Employee
Password Change
Booking
Customer
Staff Login
Login
Sign up
About Us
Home Page
Retrieve info via get requests, update via Put requests.
Put requests to both employee and Customer APIs
Get request retrieves customer info
Post request to create customer account
Get request sent to verify login
Retrieve roster, service and booking information via get requests
Get request retrieves employee info
Post requests to generate new bookings
Get request sent to verify login
Service API
Roster API
Booking API
Employee API
Customer API
Read and Write relevant information
H2 Database

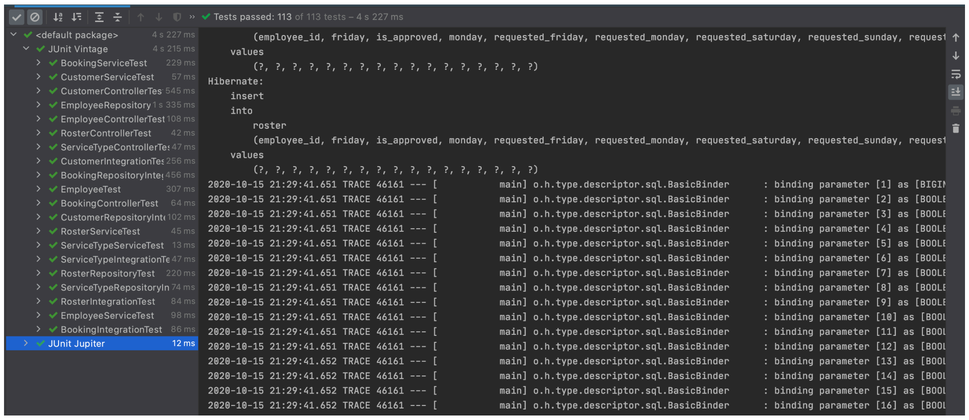
- a report describing any refactoring, including refactoring to a Design Pattern, if appropriate, as well as the "smells" that you found (i.e., purpose for refactoring)
  - **Dead Code** – Methods were removed that have become obsolete in the application due to a change in model or a bug fix where data was being displayed where it should not have been.
  - **Commenting** – The existing functions were reviewed, and extensive commenting was added to ensure each function made sense to a developer, should someone new would need to review the code.
  - **Primitive Obsession** – Removed the use of primitive data structures for fields which should not have been. This was done particularly with the employee phone number that was previously stored as an integer instead of a string.
  - **Shotgun Surgery** – Changes to data structures resulted in multiple errors thrown, particularly in the more than 50 back-end tests written, which meant a lot of changes follow the initial desired changed.
  - **Merging of methods** – Methods that were executed multiple times sequentially were merged into one call. In particular, where there were 4 or 5 *setState* calls made, these were merged into the one call which cleaned up the code repetition.
- an overview of your Gitflow organization and how often you committed
  - Our Gitflow organization were very clear, we branched out of the main branch for each new function or new task, then after implementing it we do a pull request to check it and then after confirmation, we merge it to our main branch. In this way we kept our main branch error free and our Gitflow very organized. Each of us committed as soon as an update has accrued. We tried and kept our commits as detailed as we could so If there is any error accruing during the merge, we can refer to that specific commit.
- a description of your Scrum process: how often you met, who was Scrum Master
  - Our scrum process was very organized, we meet twice a week for 2 hours each. The scrum master who is Theodore, lead the meeting, each of us state what we have done,

what each of us is planning to do and updates the whole team. not only that but we also communicated all the time through MS team, and sometimes we do hot meetings in situations where urgent questions or urgent update has been faced to discuss it immediately and to keep everyone inline and in the same page.

- a diagram of your Deployment pipeline setup: i.e,. CI/CD, steps in the pipeline, automation tools in the pipeline



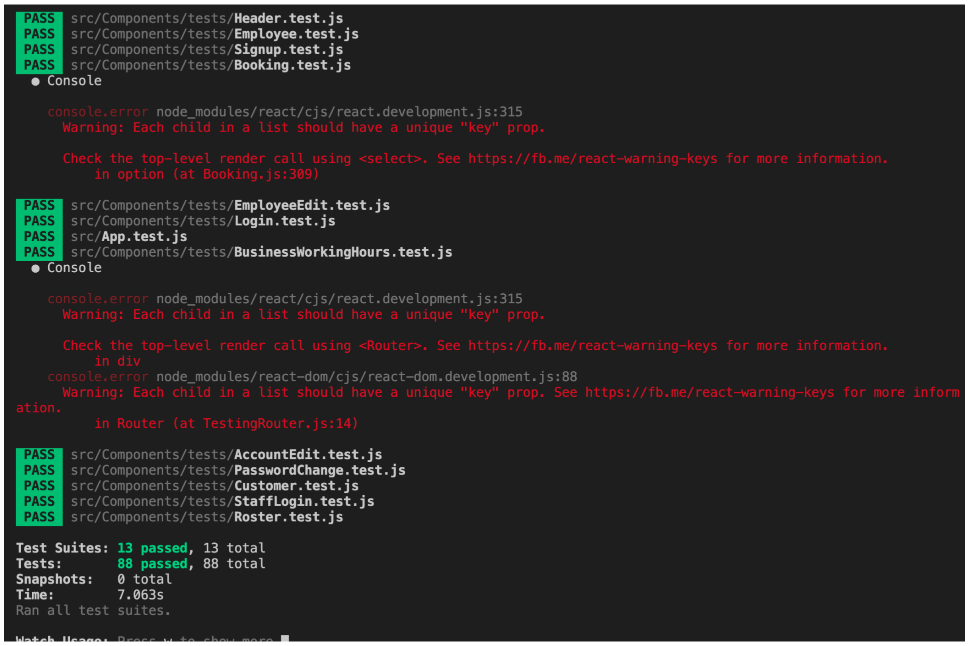
- documentation of acceptance test cases and evidence of test execution (when they were run; pass/fail status of each run)
    - Documentation of the test cases is in the test coverage folder, and the execution evidence is supplied below.



```
Tests passed: 113 of 113 tests – 4 s 227 ms

<default package>                4 s 227 ms
  JUnit Vintage                  4 s 215 ms
    BookingServiceTest           229 ms
    CustomerServiceTest           57 ms
    CustomerControllerTes        545 ms
    EmployeeRepository         1 s 335 ms
    EmployeeControllerTest       108 ms
    RosterControllerTest          42 ms
    ServiceTypeControllerTe       47 ms
    CustomerIntegrationTes       256 ms
    BookingRepositoryInte        456 ms
    EmployeeTest                 307 ms
    BookingControllerTest         64 ms
    CustomerRepositoryInte       102 ms
    RosterServiceTest             45 ms
    ServiceTypeServiceTest        13 ms
    ServiceTypeIntegrationTe      47 ms
    RosterRepositoryTest         220 ms
    ServiceTypeRepositoryIn       74 ms
    RosterIntegrationTest         84 ms
    EmployeeServiceTest           98 ms
    BookingIntegrationTest        86 ms
  JUnit Jupiter                   12 ms
```

```
        (employee_id, friday, is_approved, monday, requested_friday, requested_monday, requested_saturday, requested_sunday, request
    values
        (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
Hibernate:
    insert
    into
        roster
        (employee_id, friday, is_approved, monday, requested_friday, requested_monday, requested_saturday, requested_sunday, request
    values
        (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
2020-10-15 21:29:41.651 TRACE 46161 --- [           main] o.h.type.descriptor.sql.BasicBinder      : binding parameter [1] as [BIGIN
2020-10-15 21:29:41.651 TRACE 46161 --- [           main] o.h.type.descriptor.sql.BasicBinder      : binding parameter [2] as [BOOLE
2020-10-15 21:29:41.651 TRACE 46161 --- [           main] o.h.type.descriptor.sql.BasicBinder      : binding parameter [3] as [BOOLE
2020-10-15 21:29:41.651 TRACE 46161 --- [           main] o.h.type.descriptor.sql.BasicBinder      : binding parameter [4] as [BOOLE
2020-10-15 21:29:41.651 TRACE 46161 --- [           main] o.h.type.descriptor.sql.BasicBinder      : binding parameter [5] as [BOOLE
2020-10-15 21:29:41.651 TRACE 46161 --- [           main] o.h.type.descriptor.sql.BasicBinder      : binding parameter [6] as [BOOLE
2020-10-15 21:29:41.651 TRACE 46161 --- [           main] o.h.type.descriptor.sql.BasicBinder      : binding parameter [7] as [BOOLE
2020-10-15 21:29:41.651 TRACE 46161 --- [           main] o.h.type.descriptor.sql.BasicBinder      : binding parameter [8] as [BOOLE
2020-10-15 21:29:41.651 TRACE 46161 --- [           main] o.h.type.descriptor.sql.BasicBinder      : binding parameter [9] as [BOOLE
2020-10-15 21:29:41.651 TRACE 46161 --- [           main] o.h.type.descriptor.sql.BasicBinder      : binding parameter [10] as [BOOL
2020-10-15 21:29:41.651 TRACE 46161 --- [           main] o.h.type.descriptor.sql.BasicBinder      : binding parameter [11] as [BOOL
2020-10-15 21:29:41.651 TRACE 46161 --- [           main] o.h.type.descriptor.sql.BasicBinder      : binding parameter [12] as [BOOL
2020-10-15 21:29:41.652 TRACE 46161 --- [           main] o.h.type.descriptor.sql.BasicBinder      : binding parameter [13] as [BOOL
2020-10-15 21:29:41.652 TRACE 46161 --- [           main] o.h.type.descriptor.sql.BasicBinder      : binding parameter [14] as [BOOL
2020-10-15 21:29:41.652 TRACE 46161 --- [           main] o.h.type.descriptor.sql.BasicBinder      : binding parameter [15] as [BOOL
2020-10-15 21:29:41.652 TRACE 46161 --- [           main] o.h.type.descriptor.sql.BasicBinder      : binding parameter [16] as [BOOL
```



```
 PASS  src/Components/tests/Header.test.js
 PASS  src/Components/tests/Employee.test.js
 PASS  src/Components/tests/Signup.test.js
 PASS  src/Components/tests/Booking.test.js
  ● Console

    console.error node_modules/react/cjs/react.development.js:315
      Warning: Each child in a list should have a unique "key" prop.

      Check the top-level render call using <select>. See https://fb.me/react-warning-keys for more information.
          in option (at Booking.js:309)

 PASS  src/Components/tests/EmployeeEdit.test.js
 PASS  src/Components/tests/Login.test.js
 PASS  src/App.test.js
 PASS  src/Components/tests/BusinessWorkingHours.test.js
  ● Console

    console.error node_modules/react/cjs/react.development.js:315
      Warning: Each child in a list should have a unique "key" prop.

      Check the top-level render call using <Router>. See https://fb.me/react-warning-keys for more information.
          in div
    console.error node_modules/react-dom/cjs/react-dom.development.js:88
      Warning: Each child in a list should have a unique "key" prop. See https://fb.me/react-warning-keys for more inform
ation.
          in Router (at TestingRouter.js:14)

 PASS  src/Components/tests/AccountEdit.test.js
 PASS  src/Components/tests/PasswordChange.test.js
 PASS  src/Components/tests/Customer.test.js
 PASS  src/Components/tests/StaffLogin.test.js
 PASS  src/Components/tests/Roster.test.js

Test Suites: 13 passed, 13 total
Tests:       88 passed, 88 total
Snapshots:   0 total
Time:        7.063s
Ran all test suites.

Watch Usage: Press w to show more.
```