User story update log

Added:

As a employee, I want to be able to register myself to a business, so that I can be associated to where I work.

As a business owner, I want to see a list of all employee working for my business, so I can see who works for me.

As a employee, I want to see a list of user that made a booking with our business, so I can find their contact details if necessary.

As a user, I want a list of business so I can find out about their services

- Added user story better represent what the group has completed and aims to achieve in the sprints

Deleted:

As a new user, I want to see a noticeable "signup" button/prompt so that I don't get confused about how to sign up[1]

- As user stories focus more on functionality this user story wouldn't fit that structure as it talks about design

Summary of new Code:

- BookingException
- BusinessException
- EmployeeException
- UserException
- UserIdEception

- UserIdExceptionResponse
- CustomResponseEntityExceptionHandler

All these new files are self-created exception handling

- Booking
- Business
- Employee
- Person
- User

All these new files are object model, Person is an abstraction class for employee and user

- BookingRepository
- BusinessRepository
- EmployeeRepository
- PersonRepository
- UserRepository

All these are interface for the repository of each model object

- BookingService
- BusinessService
- EmployeeService
- MapValidationErrorService
- UserService

All these are classes that contains method for the controller to handle data in the repository and MapValidationErrorService is used to handle error of the services

- BookingController
- BusinessController
- EmployeeController
- UserController

All these classes are web mapping for the front end to call upon the necessary method to do specific task with the data

- BookingAction
- BusinessAction
- EmployeeAction
- PersonAction

These are classes used to connect with the backend and store the database into the backend

- AddBooking
- AddBusiness
- AddEmployee
- AddUser

These are classes used to format the web page to take in the input of the details of the object

- CreateEmployeeButton
- CreateUserButton
- CreateBusinessButton

These classes are used to create a button that can refactor the href

- BookingDataTable
- BusinessDataTable
- UserDataTable
- EmployeeDataTable

These classes are used to create the web page that will display all the object from the database

- BookingPost
- BusinessPost
- EmployeePost
- UserPost

These classes are used to retrieve the data from the backend database

- PostBookingButton
- PostBusinessButton
- PostEmployeeButton
- PostUserButton

These classes are used to create a button for refactoring the href to the display web page

Functionality

- Creating a booking object
- Creating a business object
- Creating a employee object
- Creating a user object
- Dsiplaying all business
- Displaying all Booking
- Displaying all employee
- Displaying all user

Testing

- BusinessServiceTest

Test the businessService class

- Business test

Test the business model class

- Person test

Test the user model class

- UserServiceTest

Test the userService class