Functional Requirements

In Scope

1. The system should display a home page with functionality such as login/sign-up, about and contact-us. The program must authenticate and authorise users based on user type/role, i.e., business owner or customer. You have three types of users: an admin user (business owner); workers, who can login and see their dashboard; customers, who want to book an appointment for a service.
2. User registration for customers: name, address, phone, username and password. Customers can register themselves and the data must be saved to the database.
3. An Admin user can login, and upon successful login, the admin is able to add/edit a new employee, add/edit working time/dates for the next month, look at the summary of past bookings (sorted date), new booking, view all workers' availability for the next 7 days.
4. A Customer can check for available days/time and services and the worker who is providing the service.
5. A customer can book a service for a specific service and date and the worker.
6. Each customer has a name, username, password, address, contact number; this data can be shown and edited through the customer profile page. You must have at least 5 customers in the db.
7. A customer must be able to cancel a booking until 48 hours before the appointment.
8. A customer must be able to display a history of their bookings.
9. Each worker must have a profile and should be able to see assigned working hours/- days and services they will provide.

User stories

Worker User Stories

As a **worker**, I want to be able to login and see the appointments I am scheduled to, with the appointment details, on the home page

Acceptance Criteria

- Criterion 1:
    - Given that appointments exist for a worker
    - When a user logs in as a worker
    - Show all appointments for that worker on their homepage.

Acceptance Tests

ID	9.3
Purpose	Test worker sees an appointment they are assigned to, on their home page.
Set Up	● User is a worker type, with an existing account. ● Appointments exist, and linked to this worker (does not matter if customer is assigned to it)
Steps	1. User logs in as a worker account.
	2. Worker arrives at home page, which shows their appointments.
Expected Result	Worker should see appointments which have not passed already (so, the start time is past the current time). They should see either the customer name on the card or 'no customer assigned' on the card.

As a **worker**, I want to be able to change my availability hours to suit my working preferences.

Acceptance Criteria

- Criterion 1:
  - Given business hours for a company (e.g. 9am - 5pm)
  - When a worker chooses a time slot
    - And the time slot is within business hours
  - Create an availability for that worker
- Criterion 2:
  - Given business hours for a company (e.g. 9am - 5pm)
  - When a worker chooses a timeslot outside of the business hours for that day
  - Do not set an availability slot for that time.

Acceptance Tests

ID	9.4
Purpose	Test worker can create an availability slot for themself
Set Up	● User is a worker type, with an existing account. ● For Tuesday 15th September, have the admin create a business hour session for 9am - 5pm for that day.
Steps	1. User logs in as a worker account.
	2. Worker arrives at home page, clicks on 'Availability' button on dashboard.
	3. Select a time slot within the business hours time zone.
Expected Result	An availability slot is created for that worker, and the admin may now assign a booking to that worker within that time slot.

ID	9.5
Purpose	Test worker cannot create an availability slot outside business hours.
Set Up	● User is a worker type, with an existing account. ● For Tuesday 15th September, have the admin create a business hour session for 9am - 5pm for that day.
Steps	1. User logs in as a worker account.
	2. Worker arrives at home page, clicks on 'Availability' button on dashboard.
	3. Select a time slot outside the business hours time zone.
Expected Result	System will not create an availability slot for that worker, and hence, it will not show up on the admin's availability view for each worker.

As a **worker**, I want to see my personal details on my profile page.

Acceptance Criteria

- Criterion 1:
    - Given a worker user logs in
        - And the login is successful
    - When a worker navigates to their profile
    - Show worker personal details on page.

Acceptance Tests

ID	9.6
Purpose	Test worker sees their personal details on the profile page.
Set Up	● User is a worker type, with an existing account.
Steps	1. User logs in as a worker account.
	2. Worker navigates to profile page.
	3. Profile details shown to worker.
Expected Result	The profile page shows the personal details for the worker, which is fetched from the backend. Personal details would include first name, last name, username, don't think any else.

As a **worker**, I want to be able to see my history of appointments.

Acceptance Criteria

- Criterion 1:
    - Given worker has appointments assigned to them in the past (i.e. before current date)
    - When a worker navigates to their appointments history page
    - Show all past appointments that were assigned to this worker.

- Criterion 2:

Acceptance Tests

ID	9.7
Purpose	Test worker can see all their past appointments on a single page.
Set Up	● User is a worker type, with an existing account. ● There are existing appointments that are already passed for the worker.
Steps	1. User logs in as a worker account.
	2. Worker navigates to their appointments history page.
	3. Worker can see all their past appointments.
Expected Result	Shows all appointments to the worker, each appointment having the time for that appointment, and what customer was assigned (can probably have no customer assigned as well, if no customer booked for that time).

Admin User Stories

As an **admin**, I want to be able to see business hours, so I can plan the monthly workload.

Acceptance Criteria

- Criterion 1:
    - When a user logs in as a admin
    - Show all business hours on the calendar as of that day

Acceptance Tests

ID	3.6
Purpose	Test admin sees business hours assigned.
Set Up	● User is an admin type, with an existing account. ● Business hours exist from today onwards, and linked to this admin.
Steps	1. User logs in as an admin account.
	2. Admin arrives at home page.
	3. Admin clicks onto the business hours tab.
Expected Result	Admin should see all business hours which have not passed already (so, the start time is past the current time).

As an **admin**, I want to be able to add business hours, so workers can see where they can add availability.

Acceptance Criteria

- Criterion 1:
    - When a user logs in as a admin
    - Show all business hours on the calendar as of that day
    - Add business hours from selectable portion of calendar

Acceptance Tests

ID	3.7
Purpose	Test admin can add business hours
Set Up	● User is an admin type, with an existing account. ● Business hours exist from today onwards, and linked to this admin.
Steps	1. User logs in as an admin account.
	2. Admin arrives at home page.
	3. Admin clicks onto the business hours tab.
	4. Admin selects a relevant portion of calendar
	5. Admin clicks ok on popup
Expected Result	Admin should see added business hours on the calendar.

As an **admin**, I want to be able to remove business hours, so I can adjust workload for the next month.

Acceptance Criteria

- Criterion 1:
    - When a user logs in as a admin
    - Show all business hours on the calendar as of that day
    - Remove business hours event from calendar

Acceptance Tests

ID	3.8
Purpose	Test admin can add business hours
Set Up	● User is an admin type, with an existing account. ● Business hours exist from today onwards, and linked to this admin.
Steps	1. User logs in as an admin account.
	2. Admin arrives at home page.
	3. Admin clicks onto the business hours tab.

	4. Admin selects an event within calendar
	5. Admin clicks ok on popup
Expected Result	Admin should see the business hours event removed from the calendar.

As an **admin**, I want to be able to add bookings for specific workers, so customers can see them.

Acceptance Criteria

- Criterion 1:
    - When a user logs in as a admin
    - Show all bookings and worker availabilities on the calendar as of that day
    - Add booking event from selectable portion of calendar constrained by worker availability

Acceptance Tests

ID	3.9
Purpose	Test admin can add booking for a specific worker
Set Up	● User is an admin type, with an existing account. ● Business hours exist for the 26th of September between 7am and 7pm. ● Worker 'waynekerr' has availability set for 26th of September between 8am to 5pm
Steps	1. User logs in as an admin account.
	2. Admin arrives at home page.
	3. Admin clicks onto the create bookings tab.
	4. Admin selects worker 'waynekerr'.
	5. Admin selects a relevant portion of calendar under worker availability shading between 10am to 12pm for the 26th of September
	6. Admin clicks ok on popup
Expected Result	Admin should see an added booking event on the calendar with null customer.

As an **admin**, I want to be able to remove a booking for a specific worker, so I can re-assign them.

Acceptance Criteria

- Criterion 1:
    - When a user logs in as a admin

- Show all bookings and worker availabilities on the calendar as of that day
- Remove existing booking event from calendar

Acceptance Tests

ID	3.10
Purpose	Test admin can remove a booking from a specific worker
Set Up	● User is an admin type, with an existing account. ● Worker 'waynekerr' has a booking for 28th of September between 8am to 10am ● No customer has booked already
Steps	1. User logs in as an admin account.
	2. Admin arrives at home page.
	3. Admin clicks onto the create bookings tab.
	4. Admin selects worker 'waynekerr'.
	5. Admin selects calendar event for the 28th of September between 8am to 10am.
	6. Admin clicks ok on popup
Expected Result	Admin should see the booking event removed on the calendar.

Definition of Done:

The definition of done that we have collectively agreed upon is that every action item on the sprint backlog is operational. This means that the features are implemented in a way that encapsulates each user story goal and their related acceptance criterias and tests. We track this progress through sprint documentation and trello management which in the spirit of this definition must be done accurately.

Scrum Documents - Sprint 1

Team name: Lemon Fruits

Peer Assessment:

Marco Barros (s3379774) Contribution: 25%

- Documentation of Sprint Planning, Sprint Review & Retro
- UI for customer booking page
- Connect Frontend to API

Adam Hoogwerf (s3719724) Contribution: 25%

- Customer booking API
- Customer profile API
- Customer history API
- Create table scripts
- Connect Frontend to API

Oskar Floeck (s3725028) Contribution: 25%

- UI for customer history
- Setup dummy data
- Connect Frontend to API

Minh Ha (s3719678) Contribution: 25%

- UI for customer dashboard
- UI for customer profile page
- Connect Frontend to API

Tools used and Links

Github repository:	remote repository link
Communication tool:	Microsoft teams
Sprint management:	Sprint 1 Trello Board and Master Trello Board

Communication Logs

Minh Ha 21/08 03:15

Oskar Floeck **Marco Barros** Adam Hoogwerf I mighta done a bit of the home page on feature/frontend/customer-home branch lol.
I think it should be dynamic (as in it's just one page, but load different parts depending on the userType), and it looks like the wireframe rn.
This is JUST an example of everything its good that it looks like the wireframe BUT if you guys find components in a g

See more

2 replies from you and Oskar

Reply

27 August 2020

Marco Barros 26/08 01:33

Minh Ha this might be stupid but how do you setState from another component? As in from the dashboard to the home

5 replies from you and Minh

Reply

Oskar Floeck 27/08 13:35

meeting in a min

Reply

Oskar Floeck 27/08 13:44

hello my aliens

Adam Hoogwerf 27/08 13:45

be there in a sec !

Reply

green started

7 replies from Minh and Oskar

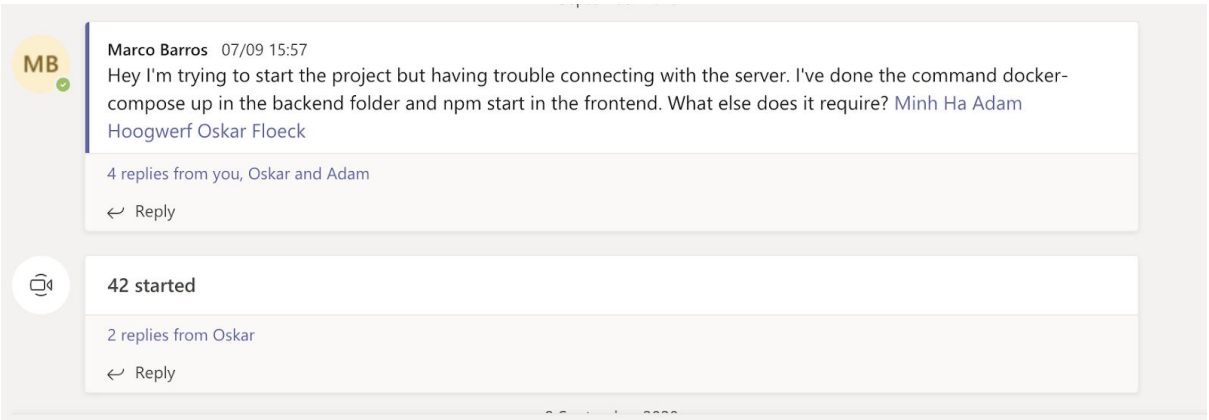
Reply

28 August 2020

Minh Ha 28/08 02:52

```
switch (userType) {
    case ("customer"):
        switch (toggleView) {
            case "Home":
                return <HomeAppointments userDetails={userDetails}/>
            case "PersonalDetails":
                return <ProfilePage/>
            case "Booking History":
                return <PastAppointments userDetails={userDetails}/>
        }
    case ("worker"):
        switch (toggleView) {
            case "Home":
                return <HomeAppointments userDetails={userDetails}/>
            case "Book Appointment":
                return <BookingPage/>
        }
```

break



Sprint Planning

Sprint: Sprint 1

Date: 17/08/2020

Team: Team 4

Scrum Master: Marco Barros (s3379774)

Product Owner: Ujj Batra

Development team: Adam Hoogwerf (s3719724), Oskar Floeck (s3725028), Minh Ha (s3719678)

Sprint goal

The goal of this sprint is to delegate major frontend and backend implementation tasks specifically for customer functionality.

Sprint Duration

2 weeks

Team Vision

This sprint we aim to extend the web application to allow a registered customer to organise a booking, cancel a booking, see their bookings history and see their details on a profile page. In order to accomplish this the following items will be placed on the sprint backlog with their corresponding story points assigned through Scrum poker.

Story Points Estimation

- UI for customer dashboard: 5 story points

We agreed as a team that since the dashboard would require a button dashboard to work with a changing view to its right called the home view, it would take up to 5 hours.

- UI for customer booking page: 3 story points

We agreed as a team that its implementation would require research into a calendar npm package therefore up to 3 hours could be taken for integration.

- UI for customer history page: 3 story points

We agreed as a team that the display of previous bookings may take up to 3 hours to refine.

- UI for customer profile page: 2 story points

We agreed as a team that the customer profile page would take up to 2 hours given it’s relative simplicity.

- Customer booking API: 3 story points

We agreed as a team that the customer booking API would take up to 3 hours because making a booking needs to conform to various constraints before it becomes valid.

- Cancel booking API: 2 story points

We agreed as a team that the cancel booking API would take up to 2 hours because it doesn't need to return anything and simply needs the removal of a booking based on id.

- Customer profile API: 1 story point

We agreed as a team that the profile API would take only up to 1 hour because of its relative simplicity since it only requires the provision of customer details.

- Customer history API: 3 story points

We agreed as a team that customer history API would take up to 3 hours because it's a little more involved where the response entity requires a list of all previous bookings where working with dates need to be exact.

- Connect frontend to API: 2 story points

We agreed as a team that connecting the frontend to the APIs would take up to 2 hours because of the various integration issues that may pop up.

Meeting Minutes

Meeting 2 - 20th of August 2020

Meeting Details

Date:	20/08/2020	
Venue:	Microsoft Teams	
Attendees:	Marco Barros	(MB)
	Adam Hoogwerf	(AH)
	Oskar Floeck	(OF)
	Minh Ha	(MH)
Apologies:	N/A	

Information/Decisions

1. Delegation of sprint backlog items.
2. Story point assignment.

Actions Items

1.	UI for customer booking page	MB	24/08/2020
2.	Customer booking API	AH	24/08/2020
3.	UI for customer history page	OF	24/08/2020
4.	UI for customer dashboard	MH	24/08/2020

Meeting 3 - 24th of August 2020

Meeting Details

Date: 24/08/2020

Venue: Microsoft Teams

Attendees: Marco Barros (MB)

Adam Hoogwerf (AH)

Oskar Floeck (OF)

Minh Ha (MH)

Ujj Batra

Apologies: N/A

Information/Decisions

1. Discussion of issues pertaining to implementation

Actions Items

1.	UI for customer booking page	MB	27/08/2020
2.	Customer history & profile API	AH	27/08/2020
3.	UI for customer history page	OF	27/08/2020
4.	UI for customer dashboard	MH	27/08/2020

Meeting 4 - 27th of August 2020

Meeting Details

Date: 27/08/2020

Venue: Microsoft Teams

Attendees:	Marco Barros	(MB)
	Adam Hoogwerf	(AH)
	Oskar Floeck	(OF)
	Minh Ha	(MH)
Apologies:	N/A	

Information/Decisions

1. Connect Frontend to API

Actions Items

1. Add items to product backlog for next sprint 07/09/2020

Sprint Review

Sprint: Sprint 1

Date: 07/09/2020

Team: Team 4

Scrum Master: Marco Barros (s3379774)

Product Owner: Ujj Batra

Development team: Adam Hoogwerf (s3719724), Oskar Floeck (s3725028), Minh Ha (s3719678)

Sprint Goals

During the sprint we planned to implement most of the customer functionalities. For the frontend this consisted of implementing the UI for customer dashboard, add booking and booking history page. For the backend this involved setting up the various APIs relevant to the above pages. Finally to synchronize the web application we connected the frontend UI with the relevant API endpoints in the backend.

Status Overview

The product owner gave us the action items as part of the sprint backlog. These weren't modified and they were largely completed within the sprint. We completed the features based on order required and importance where they were categorised as one of the

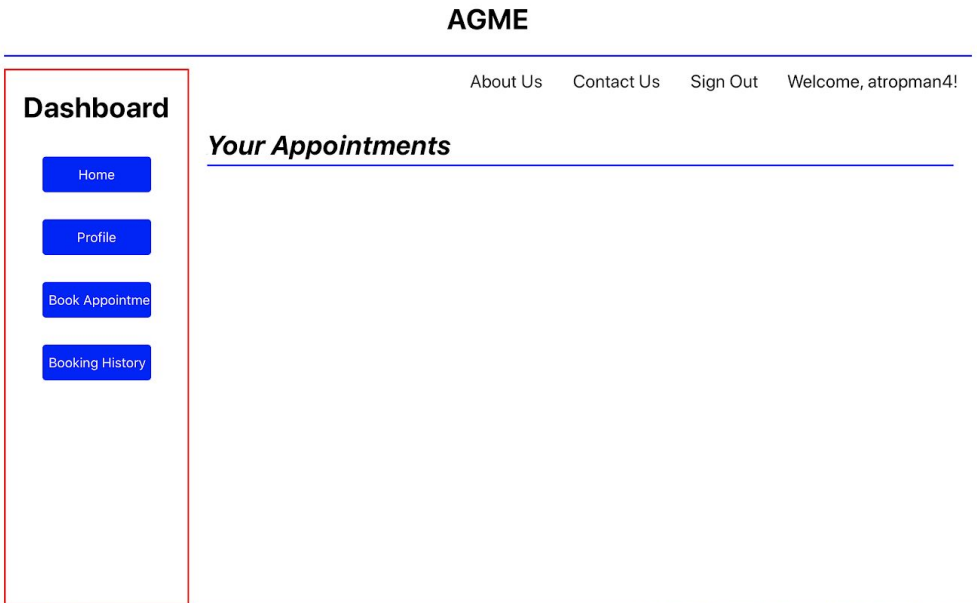
following ‘Critical’, ‘Major’ and ‘Medium’. The item ‘Cancel booking API’ has been moved to the next sprint since it wasn’t completed in the timeframe.

Screenshots in action

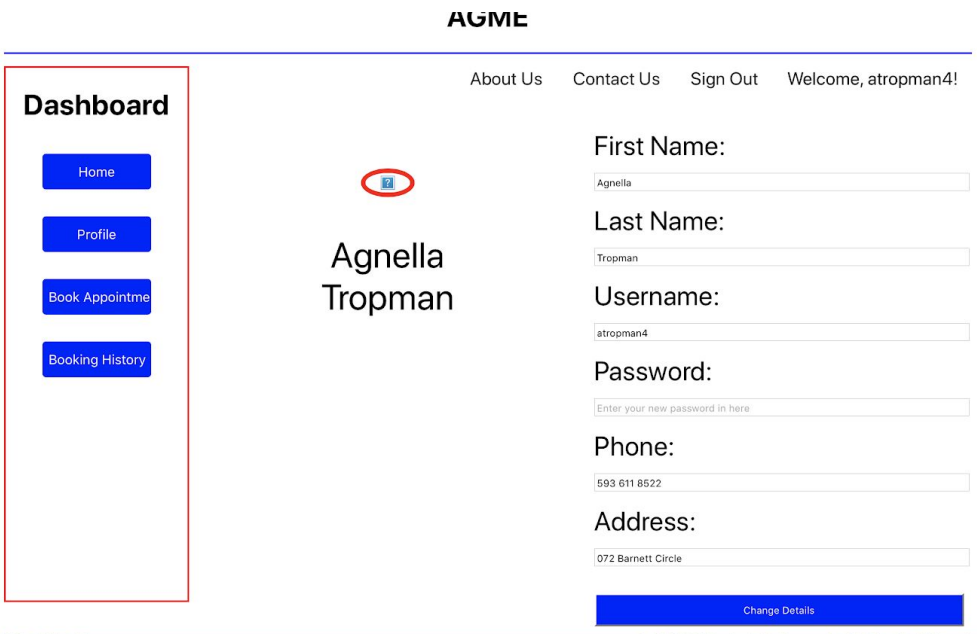
Provide some screenshots of your system working

First Run URL: *https://localhost:3000/*

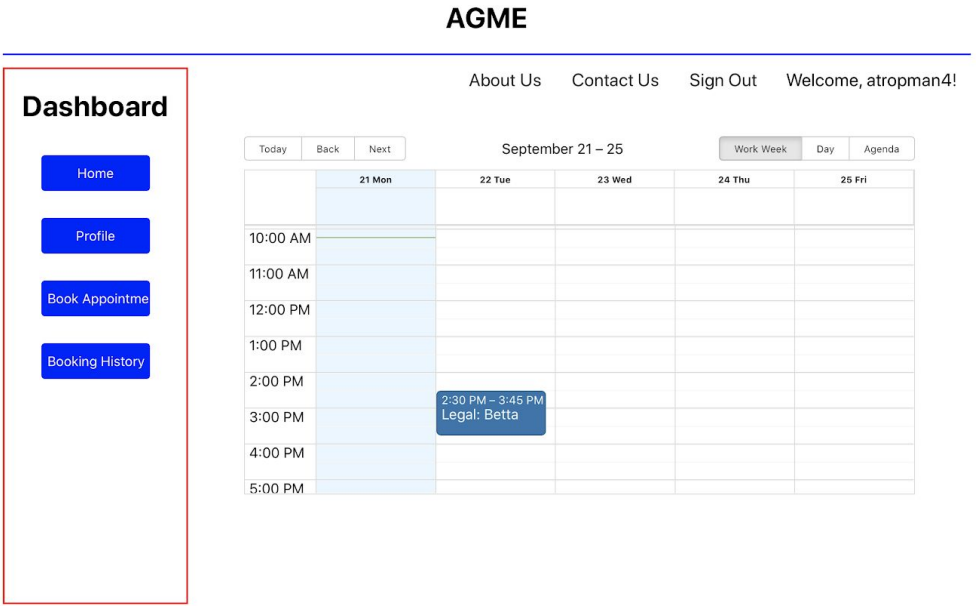
Customer Home Page:



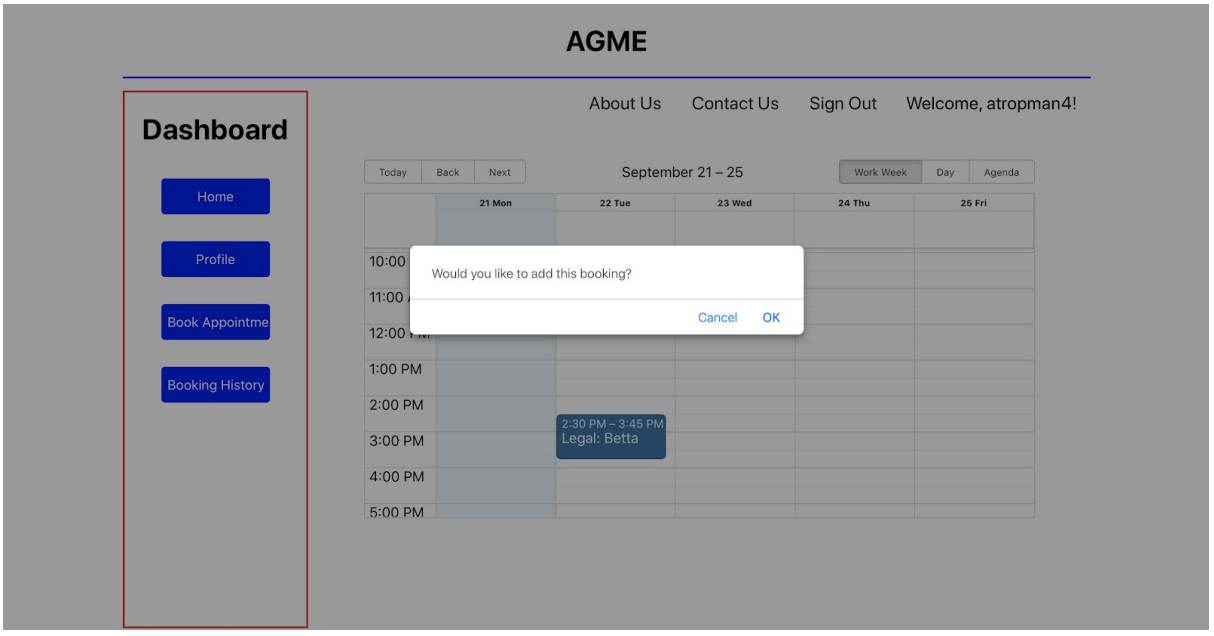
Customer Profile Page:



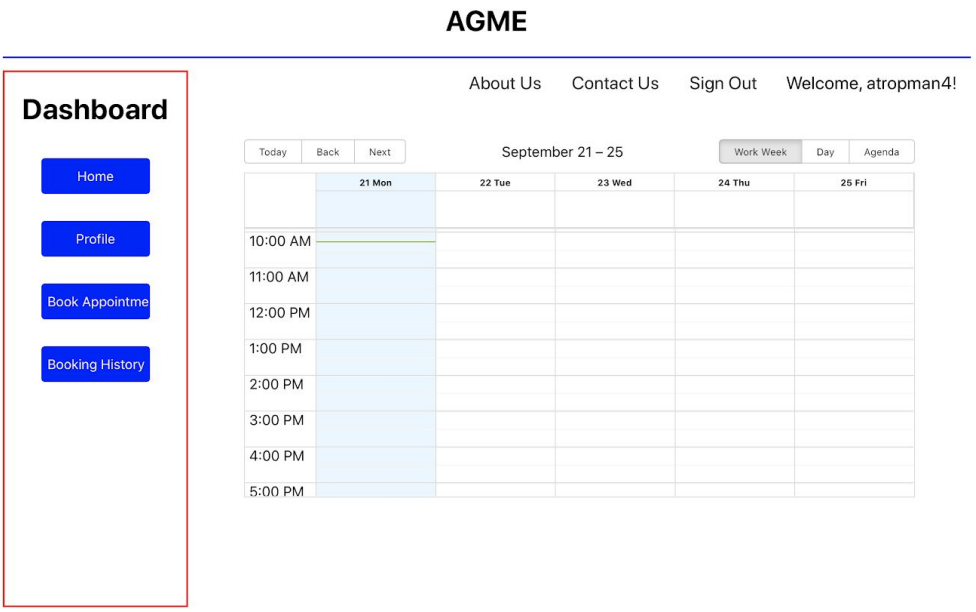
Customer Booking Page:



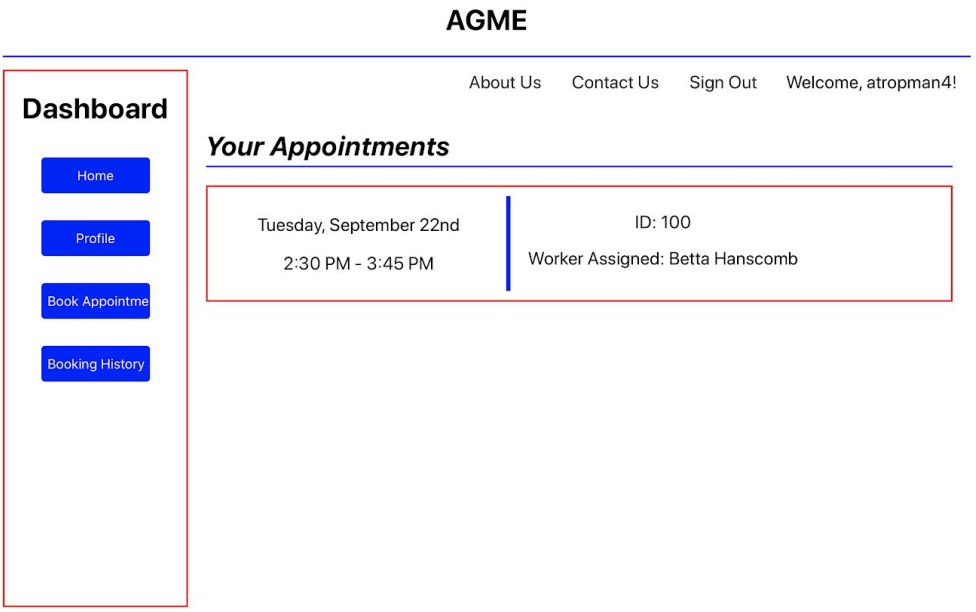
Add booking:



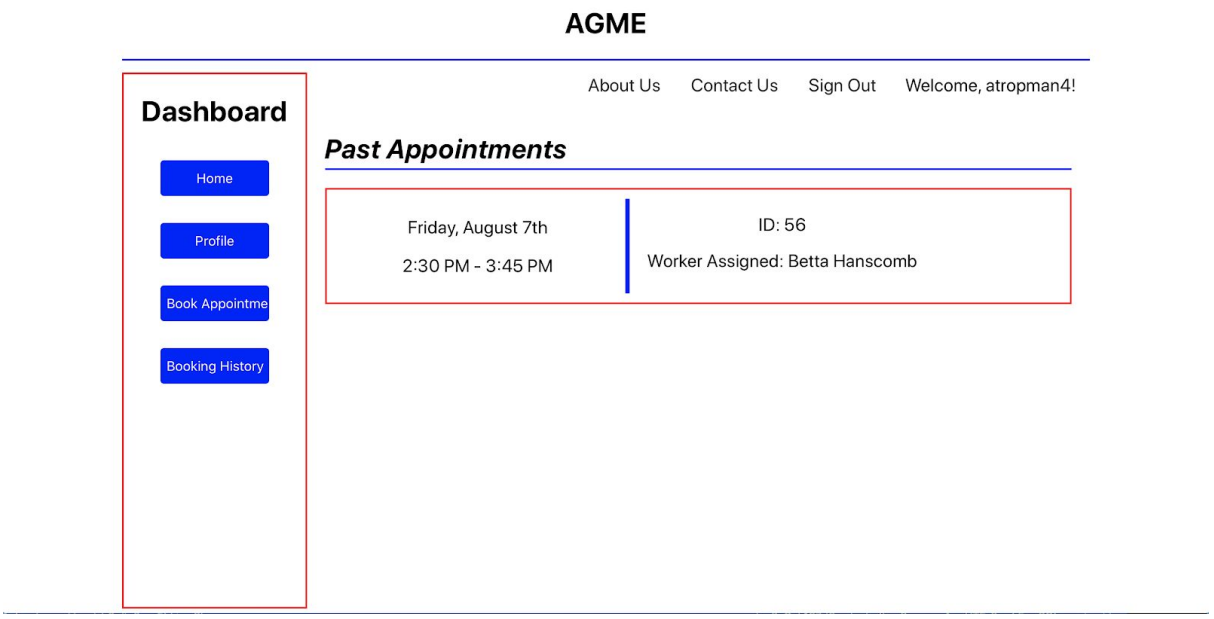
Booking page has no more available bookings:



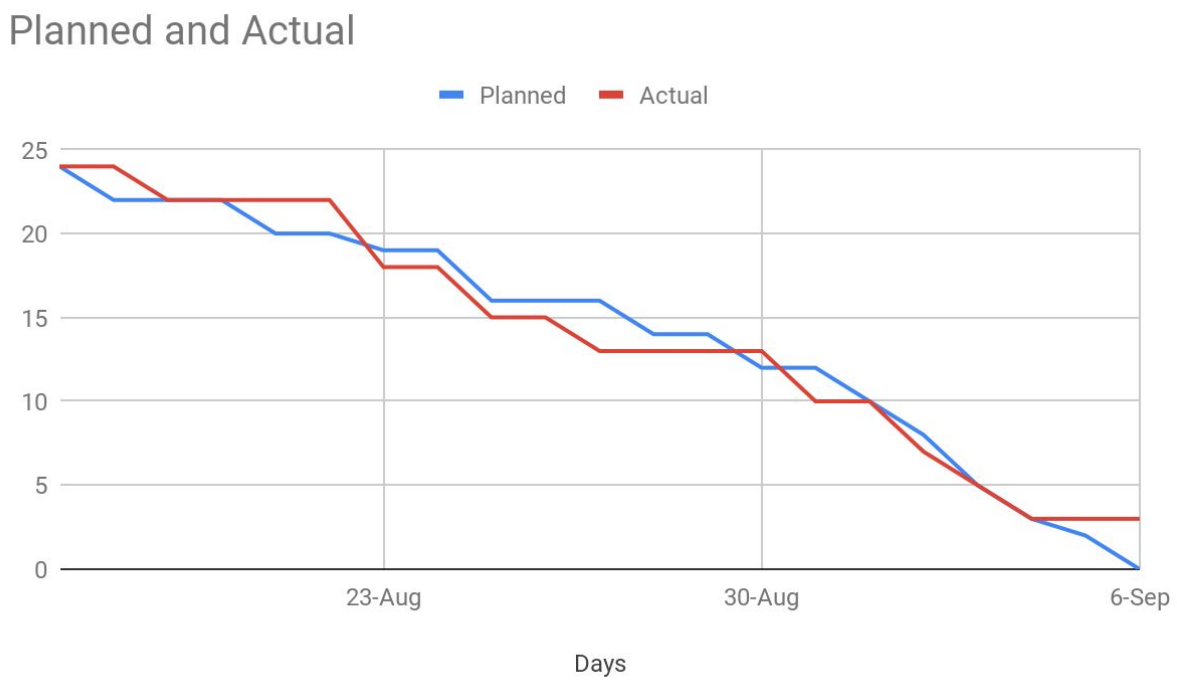
Check home page for new booking:



Customer Booking History Page:



Sprint Statistics



Task Hrs Remaining is based on the story point numbers assigned to each task.

The burndown chart shows us a graphical representation of work left to do vs. time. The green line represents what would be ideal whereas the purple line showcases the actual task hours remaining. As you can see we followed the ideal time frame quite closely.

Sprint Retro

Sprint: Sprint 1

Date: 07/09/2020

Team: Team 4

Scrum Master: Marco Barros (s3379774)

Product Owner: Ujj Batra

Development team: Adam Hoogwerf (s3719724), Oskar Floeck (s3725028), Minh Ha (s3719678)

Things that went well

- Constant communication during development for both frontend and backend.
- Supportive collaboration between team members.
- Most tasks were completed within the timeframe.

Things that could have gone better

In order to be on the same page, the frontend team must talk to the backend team more often in relation to their API work. At points frontend development hit a roadblock because of a different vision in comparison with the specification and how the APIs were set up.

Things that surprised us

Nothing truly surprised us in this sprint.

Lessons learned

To continually synchronise between the frontend and backend teams an equal understanding of the specifications involved.

Final thoughts

What are things to keep?

Constant communication within the functional teams and supportive collaboration between team members.

What are things to change?

Alignment of development vision between the functional sub-teams to ensure we're all working in the same direction.

Scrum Documents - Sprint 2

Team name: Lemon Fruits

Peer Assessment:

Marco Barros (s3379774) Contribution: 25%

- Documentation of Sprint Planning, Sprint Review & Retro
- UI for admin
- Connect admin frontend to backend
- Unit tests for frontend

Adam Hoogwerf (s3719724) Contribution: 25%

- Remove roles database & update API
- API endpoints for admin
- API endpoints for workers
- Unit test for customer controller
- Deploy on AWS

Oskar Floeck (s3725028) Contribution: 25%

- Implement CI/CD
- Cancel booking API
- Unit tests for worker controller
- Deploy on AWS

Minh Ha (s3719678) Contribution: 25%

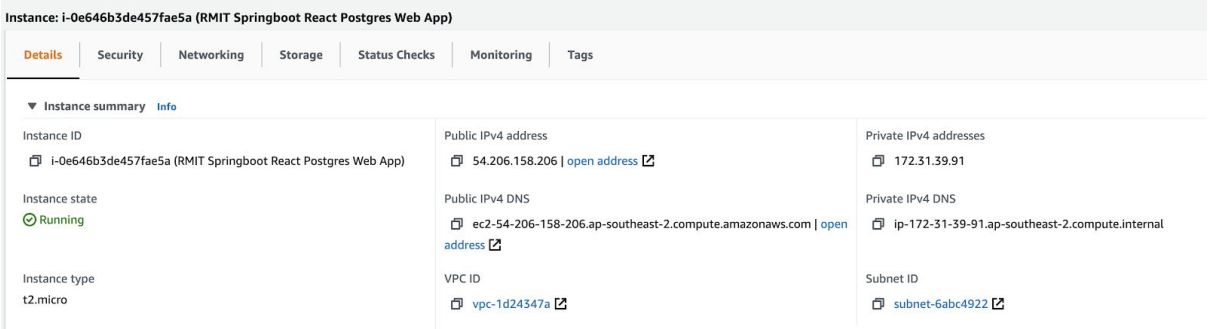
- UI for workers
- UI for admin
- Connect worker frontend to backend
- Unit tests for frontend

Tools used and Links

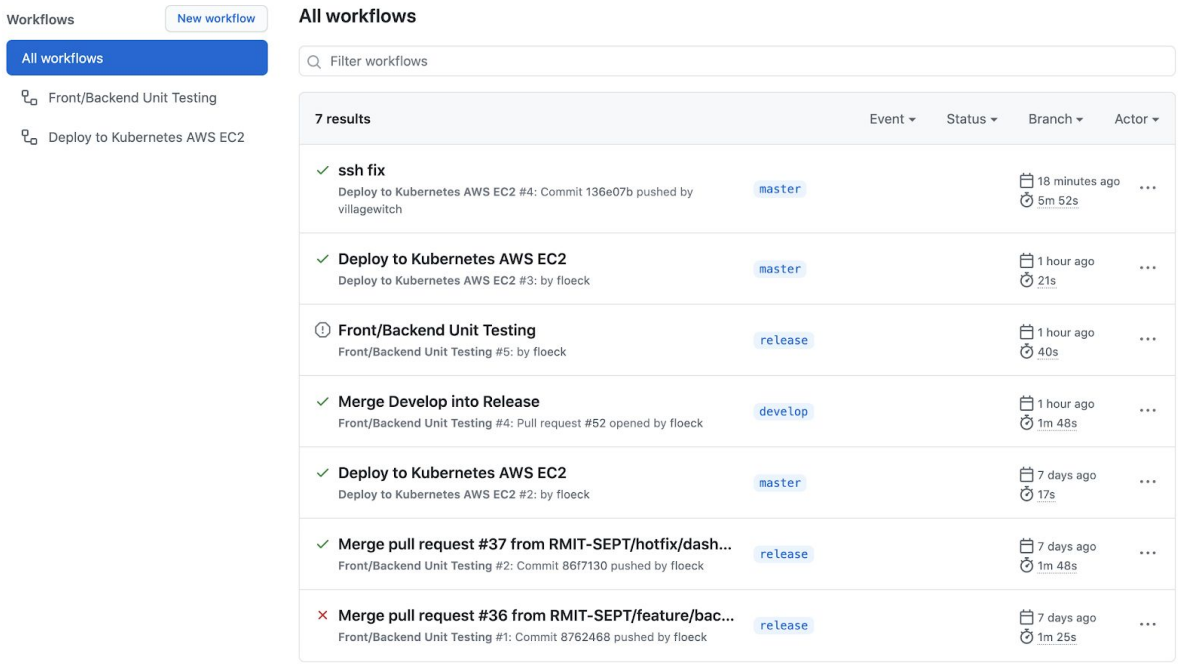
Github repository:	remote repository link
Communication tool:	Microsoft teams
Sprint management:	Sprint 2 Trello Board and Master Trello Board

AWS: [agme.company]():

AWS deployment



Github workflow



Communication Logs

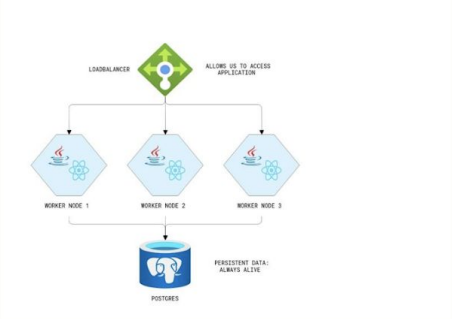
there's a snake in my boot ended:

AH MB MH

Reply

Oskar Floeck 10/09 17:59

I will explain later, but this is what the Kubernetes deployment will look like



3 replies from Oskar and Adam

Reply

MH

Minh Ha 11/09 01:50

I added two pull requests, one for profile page requesting API and one for customer users reading current and past bookings from API. Review when u guys can pls

Reply

12 September 2020

MH

Minh Ha 12/09 14:23

unit tests for progress marks tmr I think **Marco Barros** Adam Hoogwerf Oskar Floeck

6 replies from Adam, Minh and Oskar

Reply

13 September 2020

MH

Minh Ha 12/09 19:07

Marco Barros New branch I made for front end unit tests "testing/frontend/...", should be setup so you can just create the tests for whatever component

22 replies from you, Minh and Adam

Reply

Reply

Marco Barros 13/09 16:50

here

```
const HomeAppointments = ({userDetails}) => {
    const userType = userDetails.userType;

    const [appointments, setAppointments] = useState([]);

    useEffect([] => {
        const fetchData = async() => {
            // Request for customer's current bookings
            if (userType === 'customer') {
                await fetch("http://localhost:8080/api/v1/customer/view/", {
                    method: 'GET',
                    headers: {
                        'Authorization': `Bearer ${localStorage.getItem('token')}`
                    }
                }).then(response => {
                    response.json().then(array => {
                        setAppointments(array);
                    })
                })
            } else if (userType === 'worker') {
                //Do a request for worker's past appointments.
                setAppointments(null);
            }
        }
        // Have to make the async function then call it for some reason.
        fetchData();
    });

    return(
        <div id="appointmentsContainer">
            <h1> Your Appointments </h1>
            {Object.entries(appointments).map([key, value]) => {
                return <Appointment details={value}/>
            })}
        </div>
    )
}
```

19 replies from you and Minh

Reply

Marco Barros 13/09 19:42 Edited

For the 403 error with the post request, it seems like it might be an issue with webconfig.

Reply

Marco Barros 13/09 19:59

It works now changed the customer one to include all customer APIs

Config.java | SecurityConfig.java | AgmeOnlineBookingServiceApp.java

```
    JwtTokenProvider jwtTokenProvider;

    @Bean
    @Override
    public AuthenticationManager authenticationManagerBean() throws Exception {
        return super.authenticationManagerBean();
    }

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http
                .httpBasic().disable()
                .csrf().disable()
                .sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS)
                .and()
                .authorizeRequests()
                .antMatchers("/api/v1/user/login").permitAll()
                .antMatchers("/api/v1/customer/*").permitAll()
                .anyRequest().authenticated()
                .and()
                .cors().and()
                .apply(new JwtConfig(jwtTokenProvider));
    }
```

16 replies from you, Minh and Adam

Reply

14 September 2020

Adam Hoogwerf 14/09 16:28 Edited

Can everyone please go to : https://trello.com/b/9i3HnEMe/master-board And enter in any product backlog items you think we need to accomplish next sprint. **Marco Barros** Minh Ha Oskar Floeck

Trello

Organize anything, together. Trello is a collaboration tool that organizes your projects into boards. In one glance, know what's being worked on, who's working on what, and where something is in a

Oskar Floeck 15/09 14:38 1

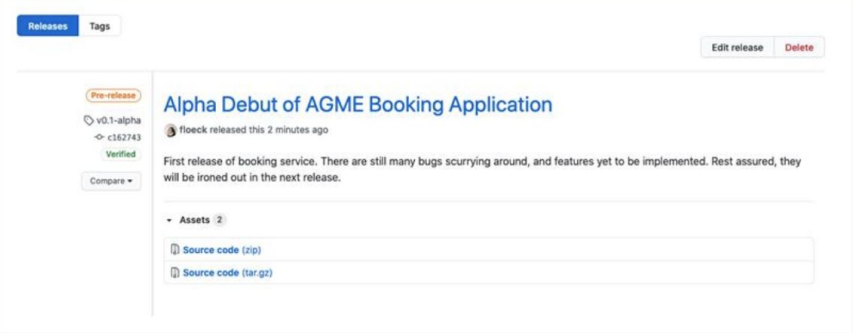
Ok I kinda already explained this but to reiterate -- Release branch is now active. Pls try keep release and develop on the same page. Any push/change to release will trigger the unit testing workflow so be mindful of that. If it passes, an automatic pull-request is made to master which will need reviewing. Once approved, another workflow is trigged on master that will deploy on the EC2 Kubernetes instance

Collapse all

Marco Barros 15/09 14:42

bloody amazing

Oskar Floeck 15/09 15:28 1



Reply

16 September 2020

Reply

16 September 2020

Minh Ha 16/09 01:10 1

Oskar Floeck Adam Hoogwerf **Marco Barros** don't forget to submit unit tests to Assignments on canvas!! (Also adam and oskar look at the convo between me and marco and give opinion when u can pls)

Oskar Floeck 16/09 08:00

No dramas cheers!

Reply

Minh Ha 15/09 01:09 1 1

Marco Barros I just did a couple of the admin pages (worker list and booking history), there's still business hours page and create bookings page left to do.

Also the really important task I think, which is kinda aids but, its to figure out what to change in our code so that users can refresh and not crash (since most pages take the 'userType/role' attribute from the previous page, but if u refresh

See more

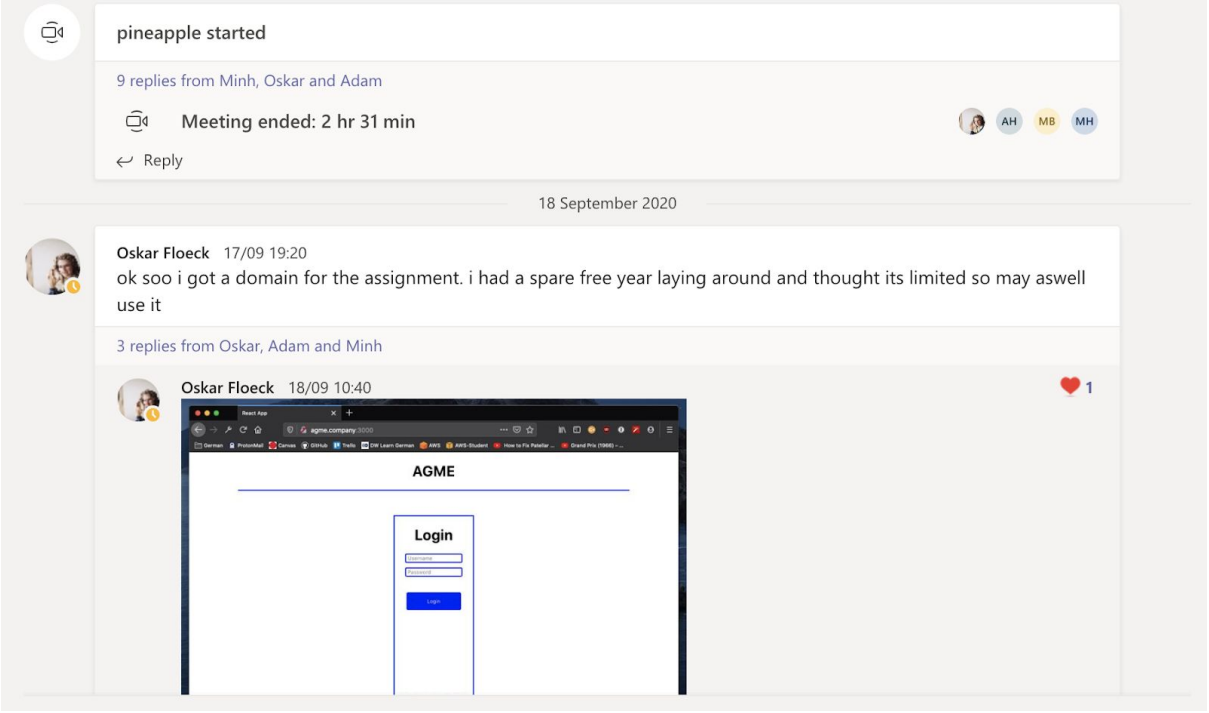
14 replies from you, Minh and Oskar

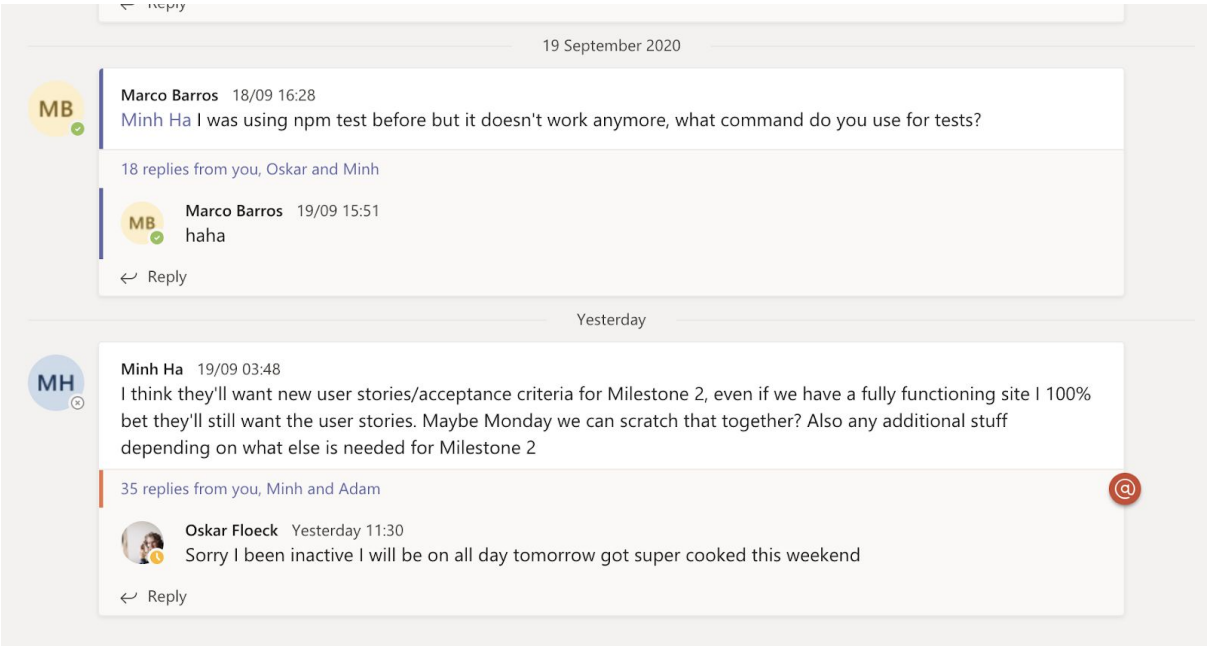
Adam Hoogwerf 16/09 14:14

Marco Barros Yep that's correct

Reply

17 September 2020





Sprint Planning

Sprint: Sprint 2

Date: 07/09/2020

Team: Team 4

Scrum Master: Marco Barros (s3379774)

Product Owner: Ujj Batra

Development team: Adam Hoogwerf (s3719724), Oskar Floeck (s3725028), Minh Ha (s3719678)

Sprint goal

The goal of this sprint is to delegate major frontend and backend implementation tasks specifically for worker and admin functionality. Furthermore we aim to deploy the website on AWS.

Sprint Duration

2 weeks

Team Vision

This sprint we aim to extend the web application to enable every user type with their respective operations. In order to accomplish this the following items will be placed on the sprint backlog with their corresponding story points assigned through Scrum poker.

Story Points Estimation

- API endpoints for admin 4 story points

We agreed as a team that it will take up to 4 hours to cover all the API endpoints for every admin operation which includes adding workers, adding bookings and business hours.

- Implement CI/CD 2 story points

We agreed as a team that investigating CI/CD and then implementing it would take up to 2 hours because once the process is understood it will be relatively easy to implement.

- API endpoints for worker 3 story points

We agreed as a team that it will take up to 3 hours to cover all the API endpoints for every worker operation. This is because the worker has less significant operations and they're less complex in comparison to the admin.

- UI for workers 4 story points

We agreed as a team that the UI for workers would take up 4 hours because of the various elements involved with respect to weekly availability, past appointments and future appointments.

- Connect worker frontend to backend 2 story points

We agreed as a team that connecting the frontend to the APIs would take up to 2 hours because of the various integration issues that may pop up.

- UI for admins 3 story points

We agreed as a team that UI for admins would take up to 3 hours because most of the pages related to admin requires repurposing the calendar object used elsewhere.

- Connect admin frontend to backend 3 story points

We agreed as a team that connecting the frontend to the APIs would take up to 2 hours because of the various integration issues that may pop up.

- Unit tests for worker controller 1 story point

We agreed as a team that unit tests for the worker controller would be relatively straightforward and should take up to 1 hour.

- Unit tests for customer controller 1 story point

We agreed as a team that unit tests for the customer controller would be relatively straightforward and should take up to 1 hour.

- Unit tests for frontend 2 story points

We agreed as a team that the combined unit tests for the frontend should take up to 2 hours but individually they'll take an hour each.

- Deploy on AWS 2 story points

We agreed as a team that deploying on AWS will take up to 2 hours because of potential deployment errors.

Meeting Minutes

Meeting 2 - 10th of September 2020

Meeting Details

Date: 10/09/2020

Venue: Microsoft Teams

Attendees:	Marco Barros	(MB)
	Adam Hoogwerf	(AH)
	Oskar Floeck	(OF)
	Minh Ha	(MH)
Apologies:	N/A	

Information/Decisions

1. Delegation of sprint backlog items.
2. Story point assignment.

Actions Items

1.	UI for admin	MB	14/09/2020
2.	API endpoints for admin	AH	14/09/2020
3.	Implement CI/CD	OF	17/09/2020
4.	UI for workers	MH	14/09/2020

Meeting 3 - 14th of September 2020

Meeting Details

Date:	14/09/2020	
Venue:	Microsoft Teams	
Attendees:	Marco Barros	(MB)
	Adam Hoogwerf	(AH)
	Oskar Floeck	(OF)
	Minh Ha	(MH)
	Ujj Batra	
Apologies:	N/A	

Information/Decisions

1. Begin implementing unit tests

2. Discussion and solution to dependency issue

Actions Items

1.	Unit tests for frontend	MB	17/09/2020
2.	Unit tests for customer controller	AH	17/09/2020
3.	Unit tests for worker controller	OF	17/09/2020
4.	Unit tests for frontend	MH	17/09/2020

Meeting 4 - 17th of September 2020

Meeting Details

Date: 17/09/2020

Venue: Microsoft Teams

Attendees: Marco Barros (MB)

Adam Hoogwerf (AH)

Oskar Floeck (OF)

Minh Ha (MH)

Apologies: N/A

Information/Decisions

1. Review unit tests
2. Assign new tasks

Actions Items

1.	Connect admin frontend to backend	MB	21/09/2020
2.	API endpoints for worker	AH	21/09/2020
3.	Deploy on AWS	OF	21/09/2020
4.	Connect worker frontend to backend	MH	21/09/2020

Sprint Review

Sprint: Sprint 2

Date: 21/09/2020

Team: Team 4

Scrum Master: Marco Barros (s3379774)

Product Owner: Ujj Batra

Development team: Adam Hoogwerf (s3719724), Oskar Floeck (s3725028), Minh Ha (s3719678)

Sprint Goals

During the sprint we planned to implement most of the worker and admin functionalities. For the frontend this consisted of implementing the UI for workers which has the following operations; see current bookings, enter their hours, show personal details and see past bookings. The UI for admin involved the following; see workers, see all bookings, make bookings and set business hours. For the backend this involved setting up the various APIs relevant to the above pages. Finally to synchronize the web application we connected the frontend UI with the relevant API endpoints in the backend for user types of worker and admin.

Status Overview

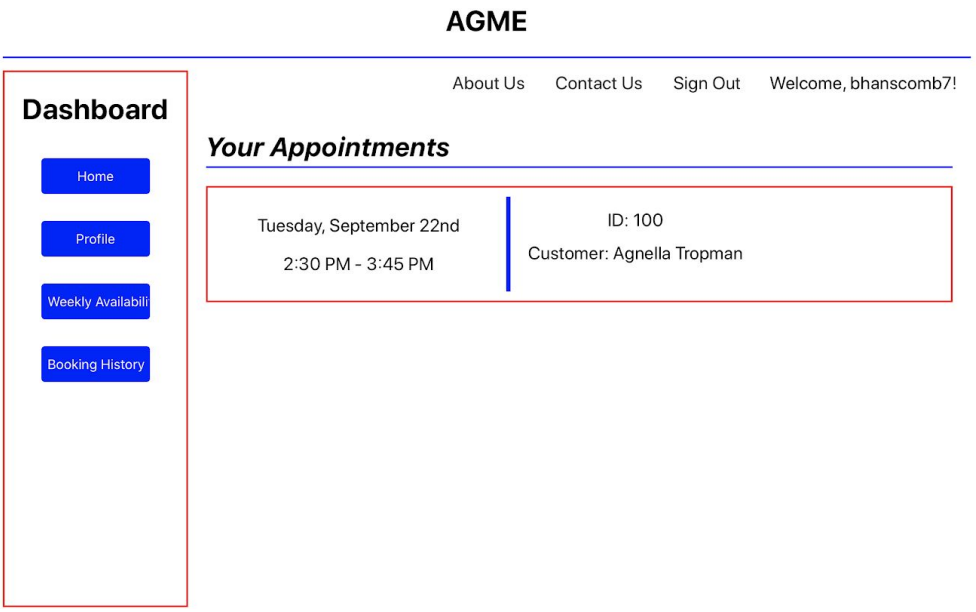
The product owner gave us the action items as part of the sprint backlog. These weren't modified and they were completed within the sprint. We completed the features based on order required and importance where they were categorised as one of the following 'Critical', 'Major' and 'Medium'. We had to push the' cancel booking API' task to the next sprint.

Screenshots in action

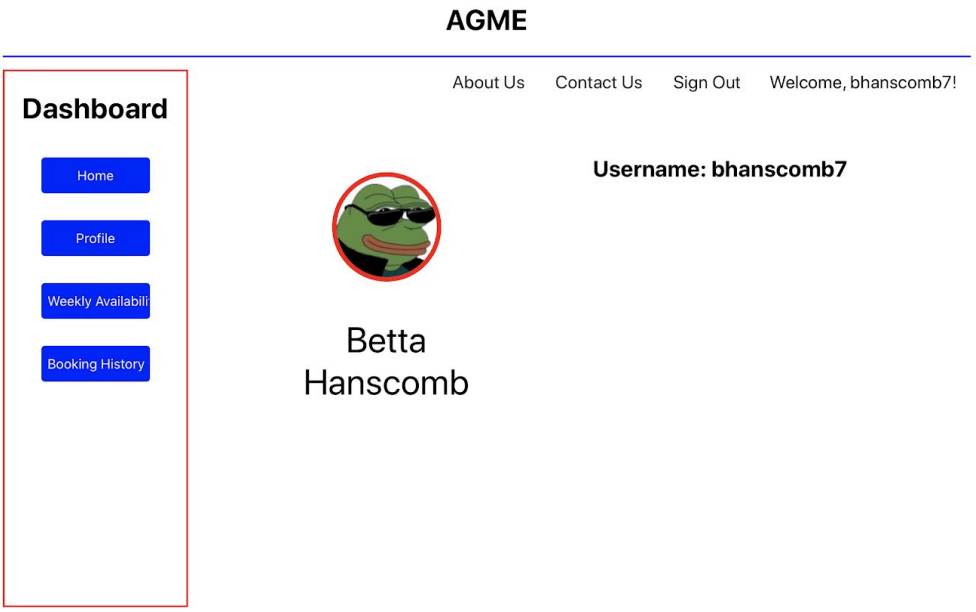
Provide some screenshots of your system working

First Run URL: *https://localhost:3000/*

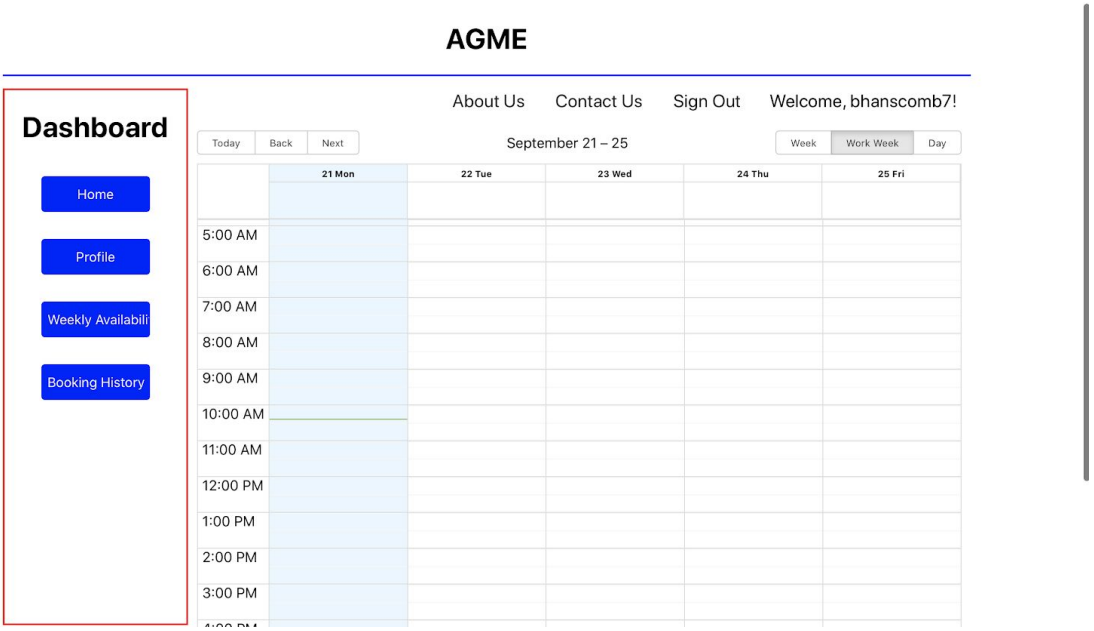
Worker home page:



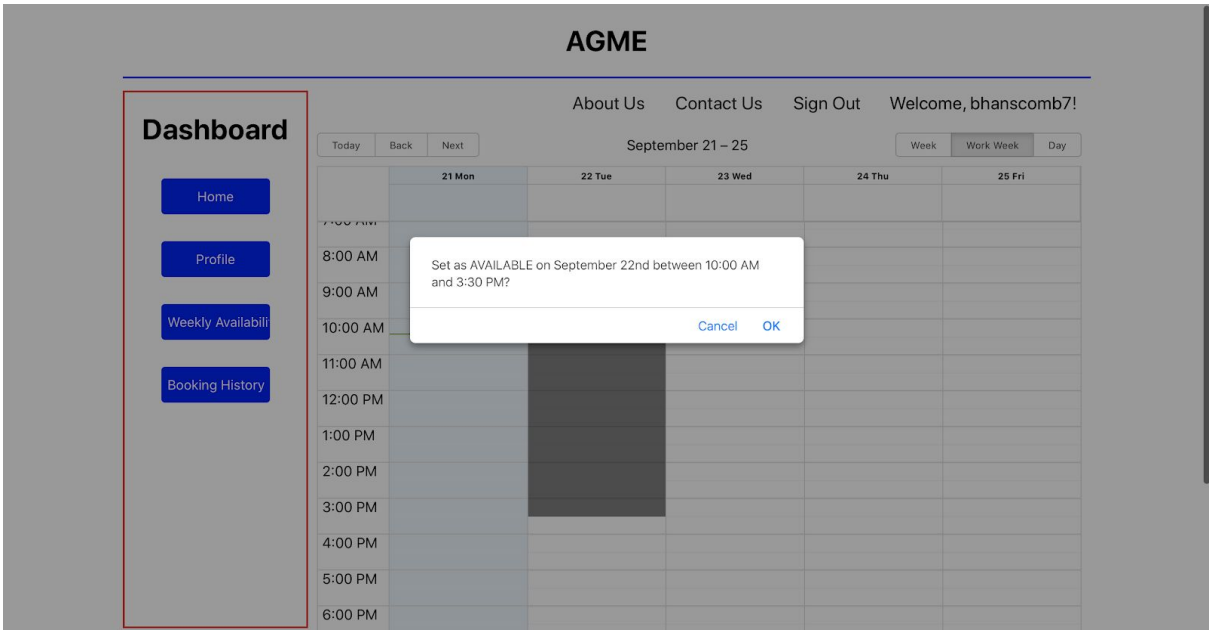
Worker profile page:

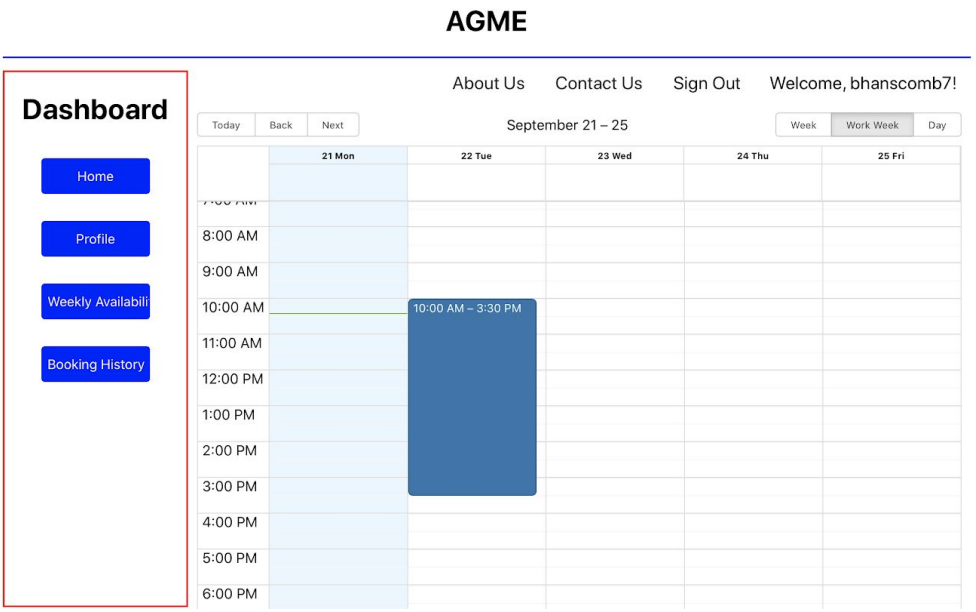


Worker weekly availability:

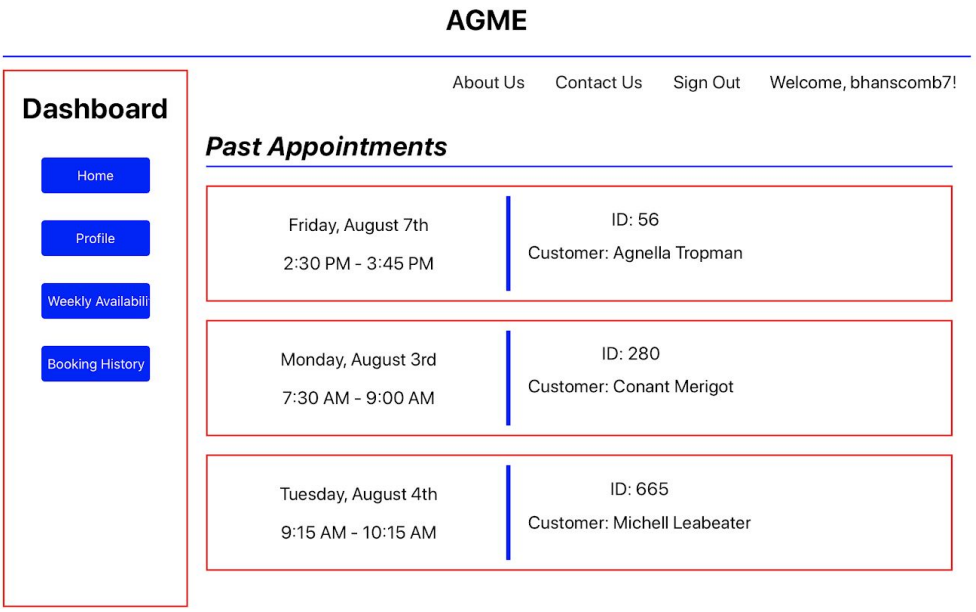


Add weekly availability:

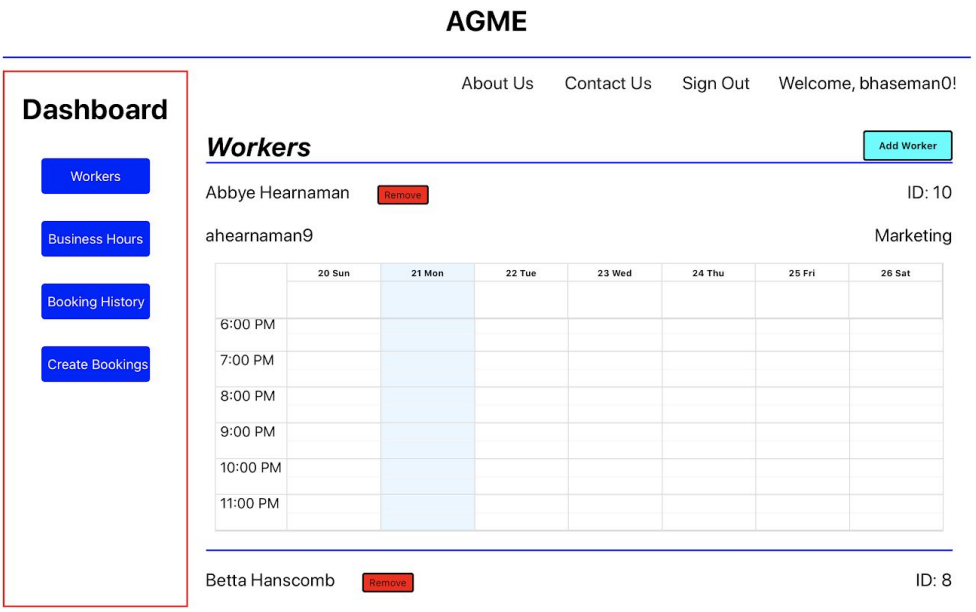




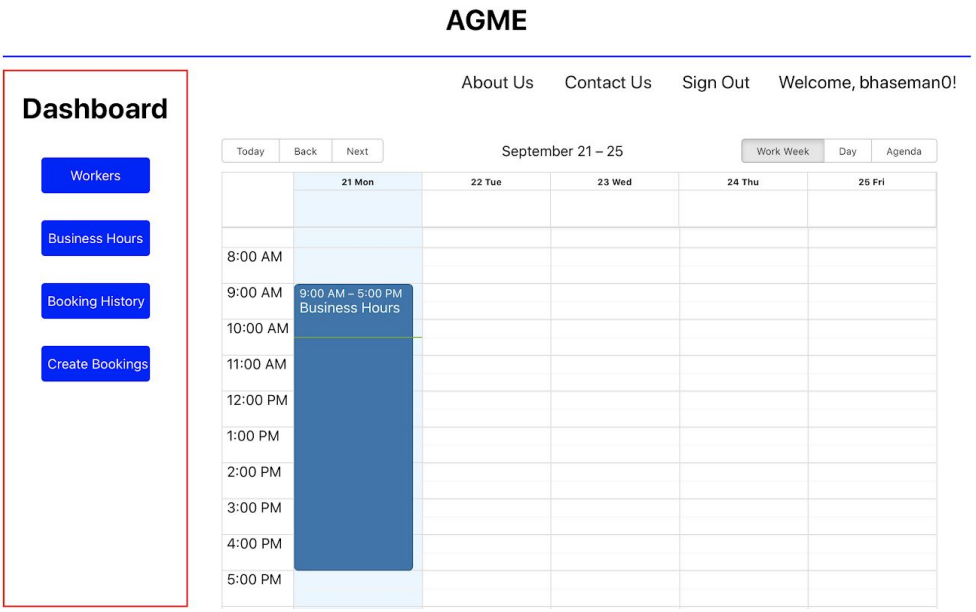
Worker past bookings page:



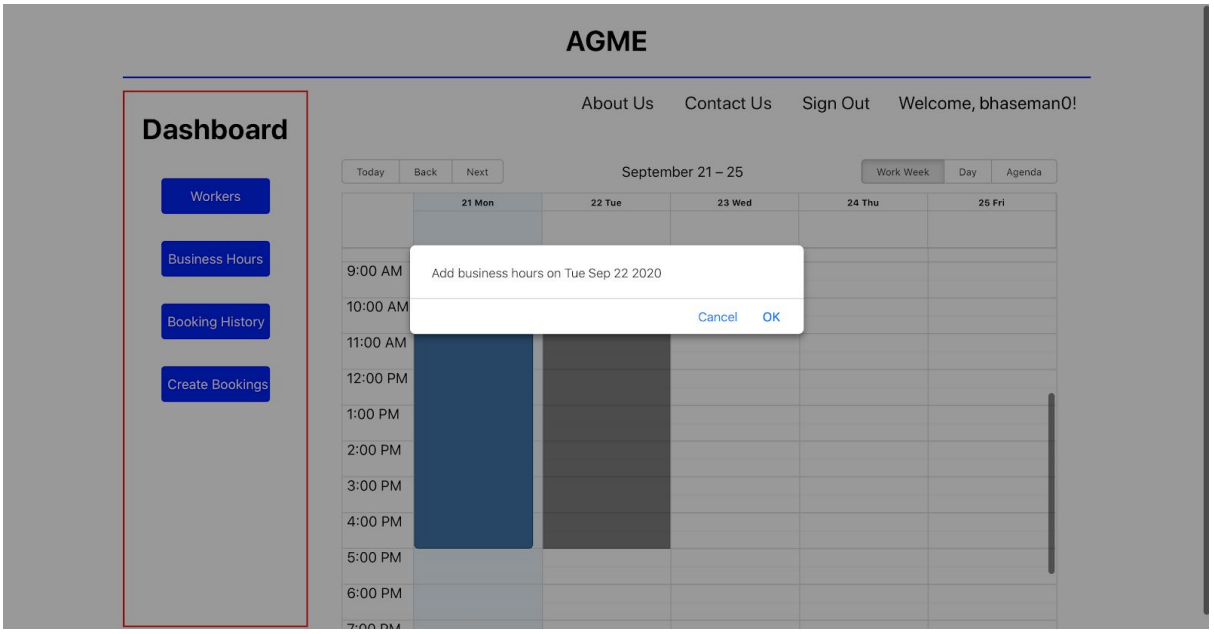
Admin Home page shows the list of workers and their availability:

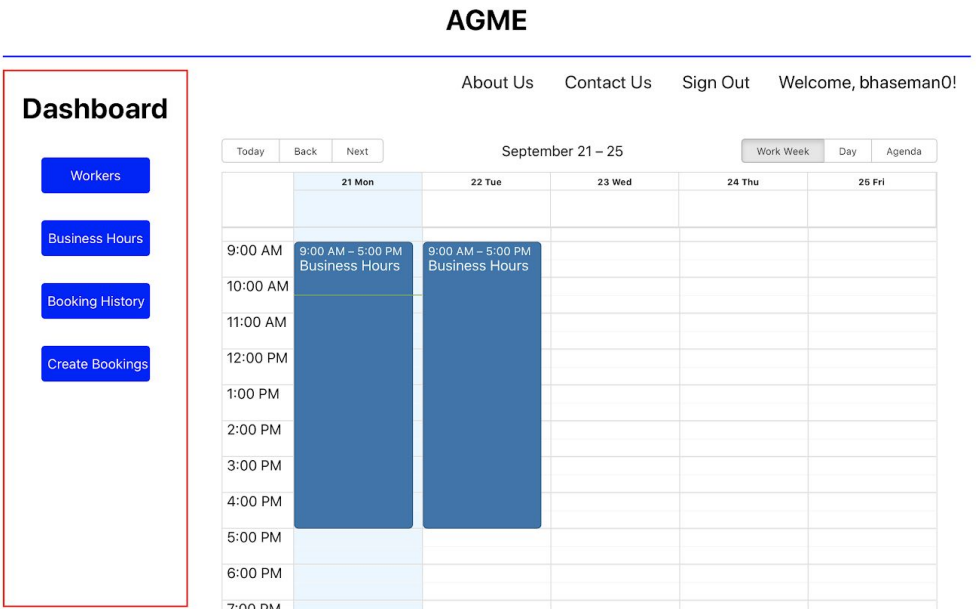


Admin current business hours:

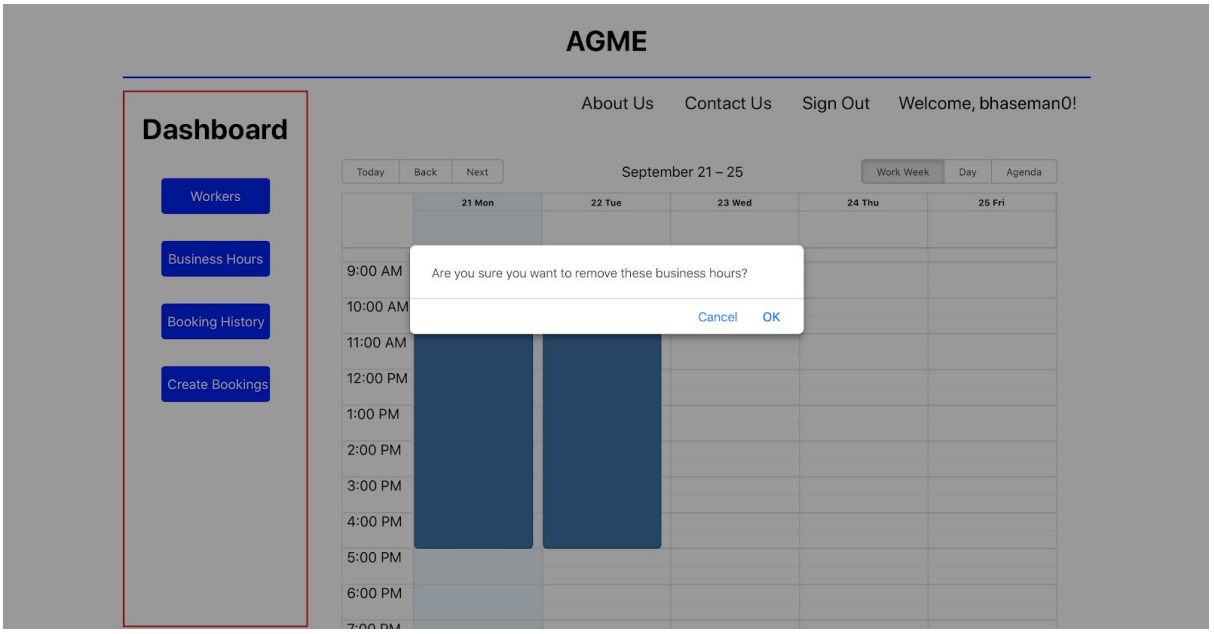


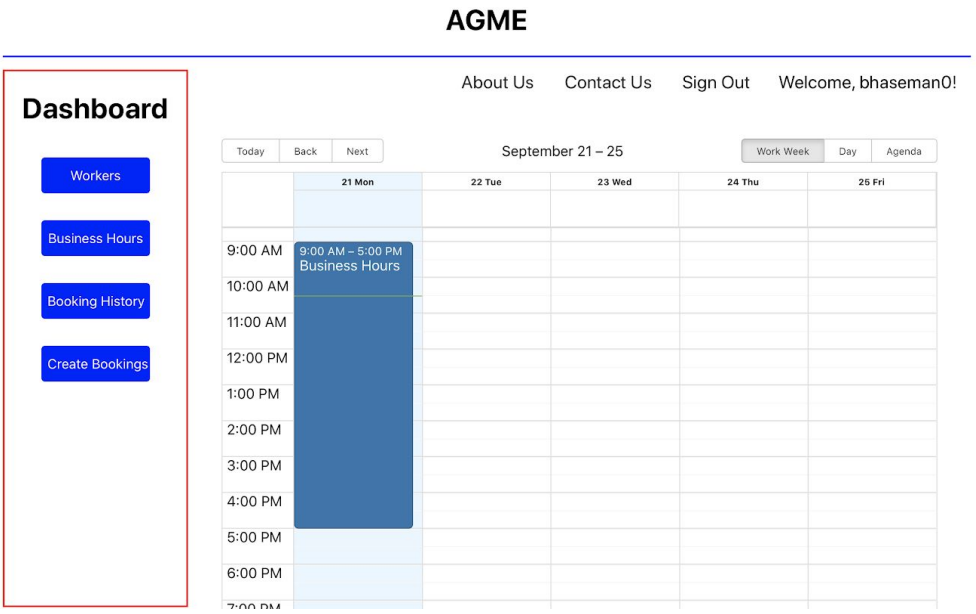
Admin set business hours:



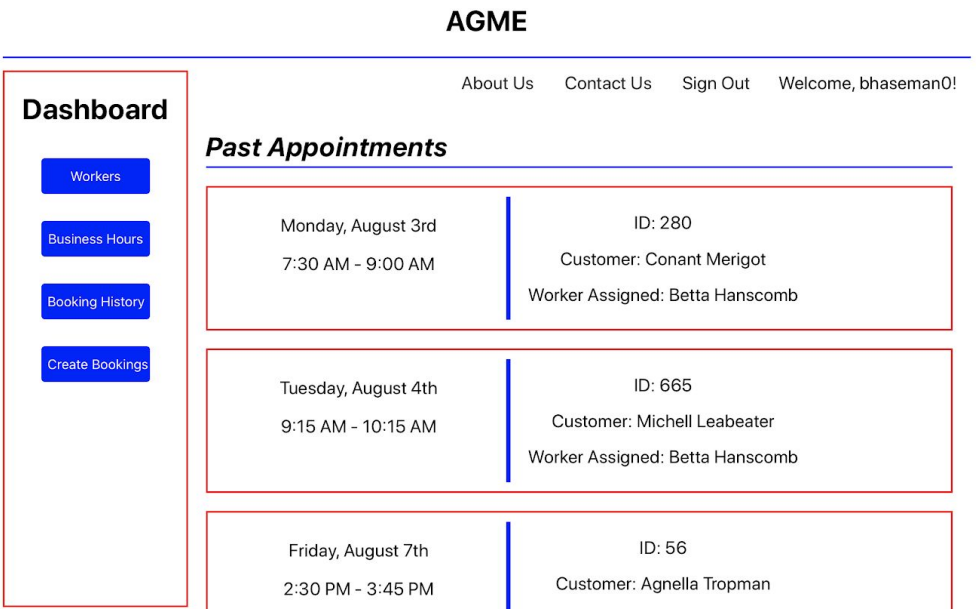


Admin remove business hours:

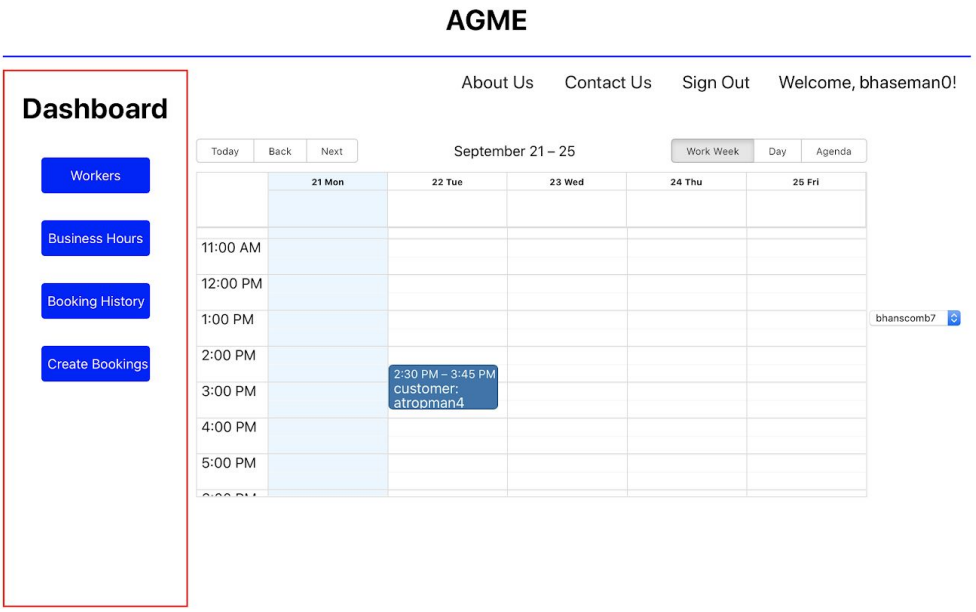


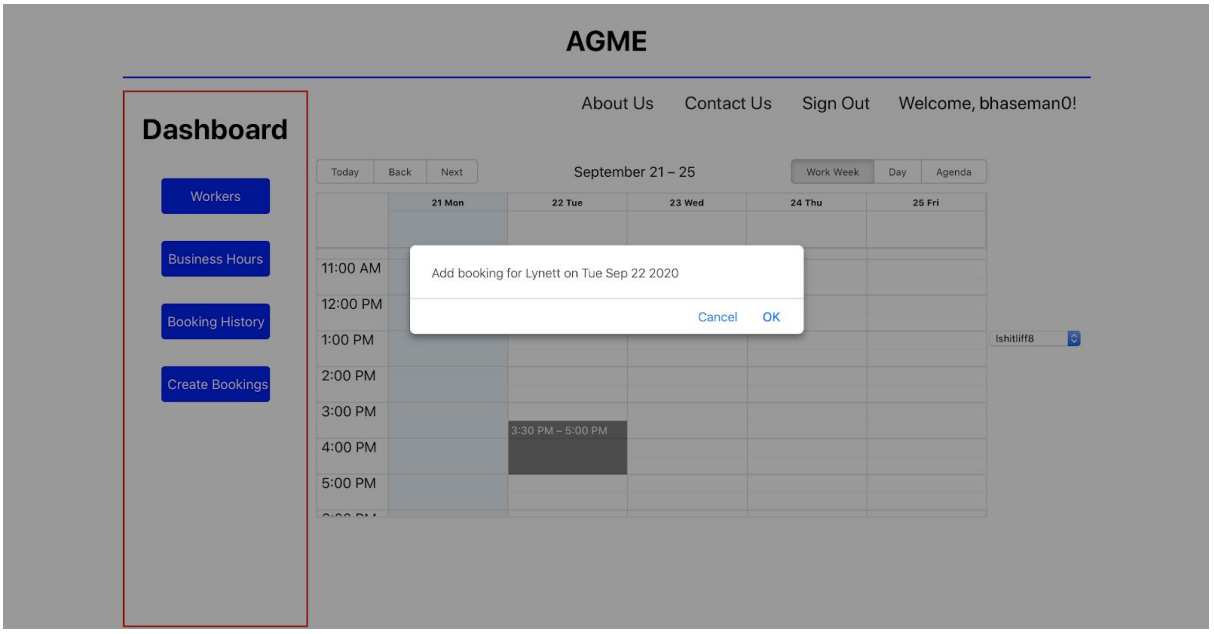


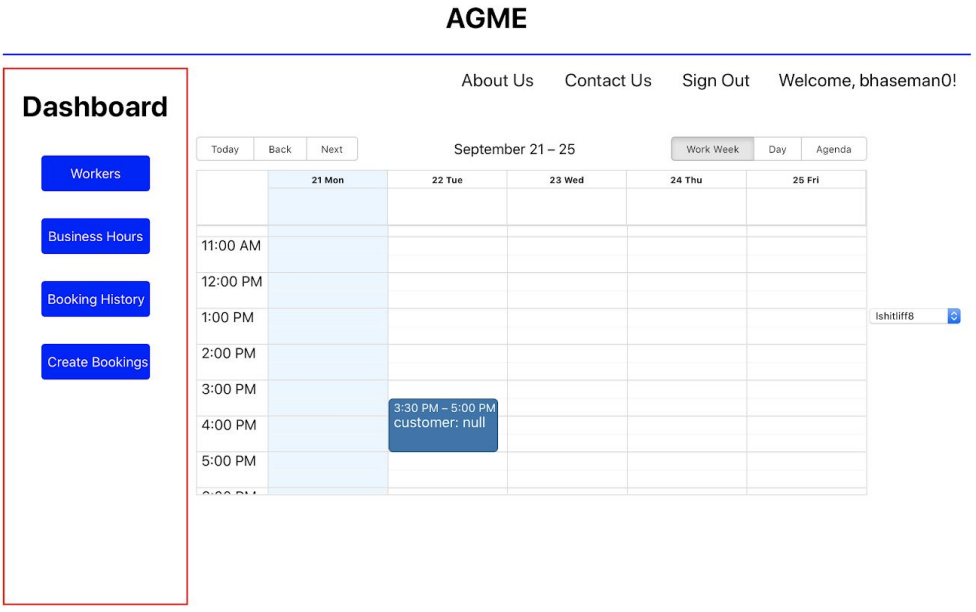
Admin past bookings:

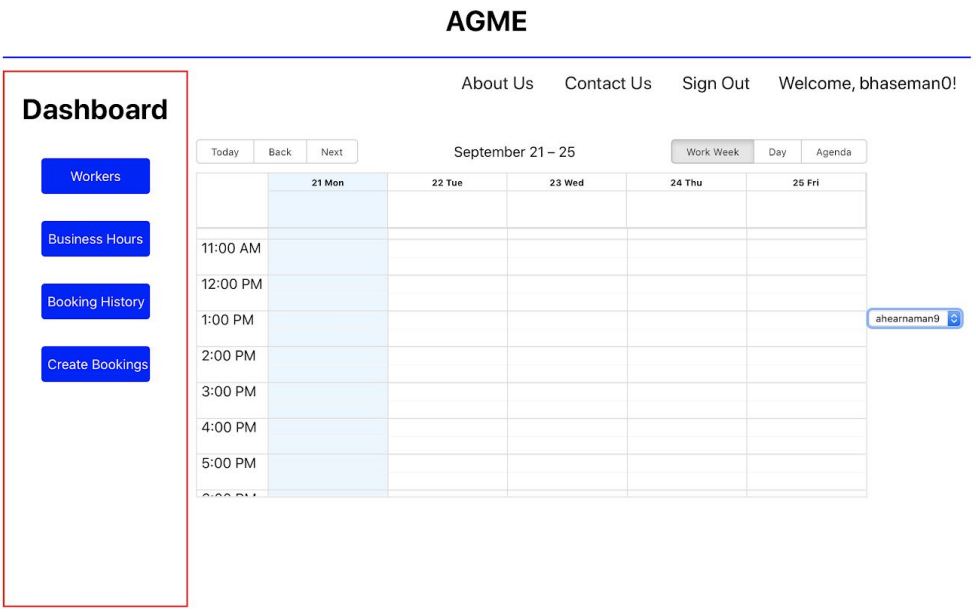


Admin create bookings with specific workers:




AGME
About Us
Contact Us
Sign Out
Welcome, bhaseman0!
Dashboard
Workers
Business Hours
Booking History
Create Bookings
Today
Back
Next
September 21 – 25
Work Week
Day
Agenda
21 Mon
22 Tue
23 Wed
24 Thu
25 Fri
11:00 AM
12:00 PM
1:00 PM
2:00 PM
3:00 PM
4:00 PM
5:00 PM
Add booking for Lynett on Tue Sep 22 2020
Cancel
OK
lshitliff8
3:30 PM – 5:00 PM


AGME
About Us
Contact Us
Sign Out
Welcome, bhaseman0!
Dashboard
Workers
Business Hours
Booking History
Create Bookings
Today
Back
Next
September 21 – 25
Work Week
Day
Agenda
21 Mon
22 Tue
23 Wed
24 Thu
25 Fri
11:00 AM
12:00 PM
1:00 PM
2:00 PM
3:00 PM
4:00 PM
5:00 PM
lshitliff8
3:30 PM – 5:00 PM
customer: null



Unit Tests

Frontend

homeview.test.js

```
FrontEnd — -zsh — 121×43
marcobarros@Marcos-MacBookPro FrontEnd % npm test homeview.test.js

> booking-app@0.1.0 test /Users/marcobarros/uni/majorproject-8-mon-17-30-4-lemonfruits/FrontEnd
> jest "homeview.test.js"

 PASS  src/__tests__/homeview.test.js
  ✓ appointment component renders details correctly (229ms)
  ✓ profile page renders user details correctly (5ms)
  ✓ booking page renders without crashing (1ms)
  ✓ home appointments component renders without crashing
  ✓ past appointments component renders without crashing (1ms)


Test Suites: 1 passed, 1 total
Tests:       5 passed, 5 total
Snapshots:   0 total
Time:        6.82s, estimated 17s
Ran all test suites matching /homeview.test.js/i.
```

dashboard.test.js

```
MINGW64:/c/Users/DucH/OneDrive/Desktop/Uni Projects/SEPT/majorproje...
remote: Resolving deltas: 100% (9/9), completed with 9 local objects.
To https://github.com/RMIT-SEPT/majorproject-8-mon-17-30-4-lemonfruits.git
   6946615..8c2640e  develop -> develop

DucH@DESKTOP-OCUNGIL MINGW64 ~/OneDrive/Desktop/Uni Projects/SEPT/majorproject-8
-mon-17-30-4-lemonfruits/FrontEnd (develop)
$ jest "dashboard.test.js"
 PASS  src/__tests__/dashboard.test.js (7.976s)
  √ renders without crashing (5ms)
  √ renders with correct buttons for customer type (14ms)
  √ renders with correct buttons for worker type (8ms)
  √ renders with correct buttons for admin type (10ms)
  √ clicking profile button (for any user type) will correctly push path onto hi
story with the current state (userDetails) (7ms)

Test Suites: 1 passed, 1 total
Tests:       5 passed, 5 total
Snapshots:   0 total
Time:        12.602s
Ran all test suites matching /dashboard.test.js/i.

DucH@DESKTOP-OCUNGIL MINGW64 ~/OneDrive/Desktop/Uni Projects/SEPT/majorproject-8
-mon-17-30-4-lemonfruits/FrontEnd (develop)
$
```

Backend

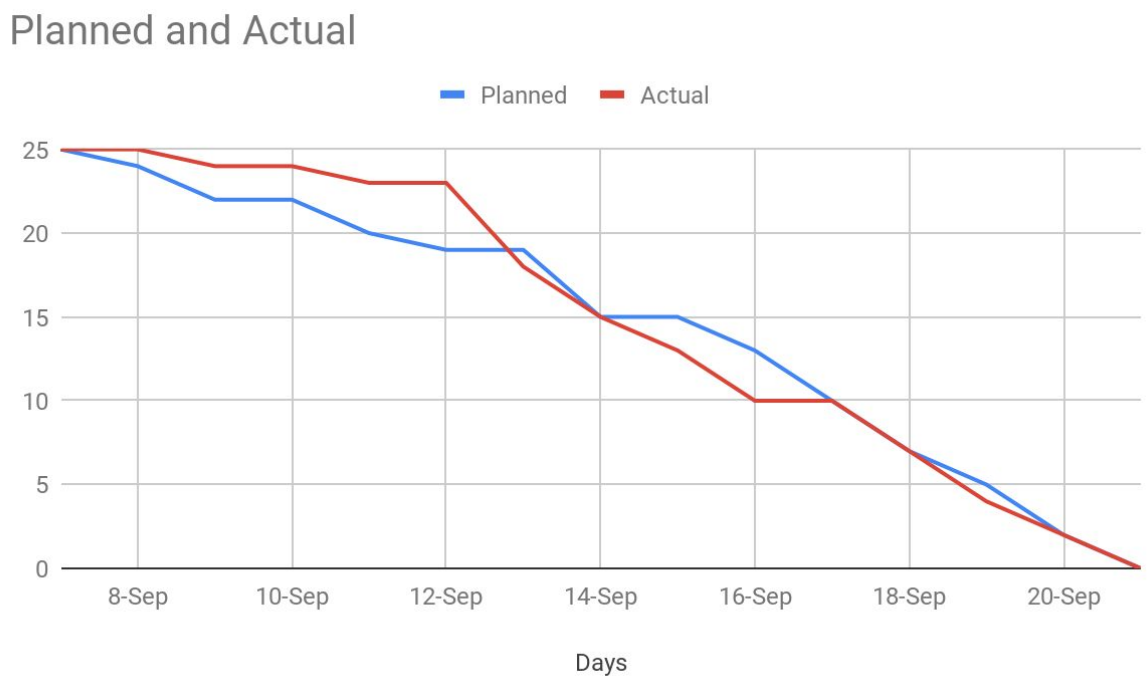
Customer Controller

▼ ✔ Test Results	557 ms
▼ ✔ CustomerControllerTest	557 ms
✔ signUpInvalidCustomerTest()	516 ms
✔ changeProfileInvalidUsername()	4 ms
✔ changeProfile()	5 ms
✔ getAvailabilities()	19 ms
✔ getCurrentBookingsTest()	2 ms
✔ getSomeCurrentBookingsTest()	1 ms
✔ signUpCustomerTest()	10 ms

Worker Controller

Test Results	550 ms
WorkerControllerTest	550 ms
removeAvailabilityTest()	517 ms
createInvalidAvailabilityTest()	8 ms
createWorkerAvailabilityTestInsideBusinessHours()	6 ms
removeMissingAvailabilityTest()	5 ms
createWorkerAvailabilityTestIncludingInsideBusiness	3 ms
createOverlappingAvailabilityTest()	3 ms
createWorkerAvailabilityTestOutsideBusinessHours()	4 ms
removeNotOwnedAvailabilityTest()	4 ms

Sprint Statistics



Task Hrs Remaining is based on the story point numbers assigned to each task.

The burndown chart shows us a graphical representation of work left to do vs. time. The green line represents what would be ideal whereas the purple line showcases the actual task hours remaining. As you can see we followed the ideal time frame quite closely.

Sprint Retro

Sprint: Sprint 2

Date: 07/09/2020

Team: Team 4

Scrum Master: Marco Barros (s3379774)

Product Owner: Ujj Batra

Development team: Adam Hoogwerf (s3719724), Oskar Floeck (s3725028), Minh Ha (s3719678)

Things that went well

- Constant communication during development for both frontend and backend.

- Great communication between frontend and backend specifically around the specifications.
- Supportive collaboration between team members.
- Most tasks were completed within the timeframe.
- Error handling for many parts of the product, to enhance user experience.

Things that could have gone better

Task delegation specifically for ‘cancel booking API’ implementation. The task was transferred from one team member to another and due to other higher priorities it was never completed during this sprint.

Communication was very constant and thorough which was good, but missed some details in some areas which led to certain functionalities being moved to next sprint (e.g. delete worker is moved to next sprint).

Things that surprised us

There was a dependency issue related to the jest package within the node modules for the frontend which was unusually annoying to fix.

On top of this, our package.json and package-lock.json was different for some people at one stage, which resulted in not being able to run the product for some. Fixed it by keeping the package.json and package-lock.json the same on the branch and running locally.

Lessons learned

Could try to improve task story point estimation and review priorities in place for assigned tasks.

Re-read the specifications and double checking we’re making the base endpoints / pages and noting them to not miss functionality and have a nasty surprise (re: ‘Things that could have gone better’)

Final thoughts

What are things to keep?

- Communication in and between the frontend and backend sub-teams.
- Supportive assistance when team members require clarification.
- Our productivity has been consistently high throughout the sprint.

What are things to change?

- Review task priorities throughout the sprint.
- Check each item in the backlog is being accounted for.