Implementation report
Table of Contents

1. Introduction

This software was implemented using the MERN stack using JetBrains WebStorm IDE.

All the code is written in JavaScript. This prevented context switching and bugs.

- M – Mongo
- E – Express
- R – React
- N – Node

Database: MongoDB/ Mongoose

I have used Mongoose to interact with my MongoDB cluster. The main advantage I gain from using MongoDB is the multiple instanced clusters, which stores the data in multiple data centers and ensures data integrity and availability.

I have used my personal MongoDB cluster for hosting a dummy database with information. See below for key. This key is currently being used in the project file. Due to confidentially reasons, I cannot hook the application the real database as it contains sensitive disabled user's information.

mongodb+srv://root:vQypmyseabzRknlu@utp-holder-twb0u.azure.mongodb.net/test?retryWrites=true&w=majority

2. Directory tree and Run-time components

2.1 Frameworks & API

GraphQL

Moreover, I have exposed the frontend with GraphQL queries to fetch only the data that is needed for a React component. This decreased load time for the web application.

Cron & Node mailer

I'm using Cron to schedule email reminders. Cron is set to execute at 12:00 every day where it checks all events records to see if there are any upcoming events. If so, the planner and testers are reminded.

Canvas JS

I'm using Canvas JS to render the tester verified proportion, sales revenue and sales forecast data.

Google API Maps Wrapper

I'm using Maps API to render a map based on tester address. This address is fetched from the tester record in the database collection. This address is encoded using a geocoder API request and the map with labels for testers are generated with their names.

2.1.1 Full list of frontend dependencies at runtime:

See **package.json** in the project for a more detailed view.

NOTE: Use **npm install** to install the listed dependencies if you are using the file without node_modules pre-installed. **This will be located in project file/node_modules and project file/frontend/node_modules.**

```
"dependencies": {
  "chart.js": "^1.1.1",
  "google-maps-react": "^2.0.6",
  "react": "16.8.6",
  "react-chartjs": "^1.2.0",
  "react-dom": "16.8.6",
  "react-router-dom": "^4.3.1",
  "react-scripts": "^3.4.1",
  "react-table": "6.10.0"
},
```

2.1.2 Full list of backend dependencies at runtime:

See package.json in event for a more detailed view.

```
"dependencies": {
  "bcryptjs": "^2.4.3",
  "body-parser": "^1.18.3",
  "dataloader": "^1.4.0",
  "express": "^4.16.4",
  "express-graphql": "^0.7.1",
  "google-maps-react": "^2.0.6",
  "graphql": "^14.0.2",
  "jsonwebtoken": "^8.4.0",
  "mongoose": "^5.9.18",
  "node-cron": "^2.0.3",
  "node-fetch": "^2.6.0",
  "nodemailer": "^6.4.10",
  "react-table": "6.10.0"
},
```

Appendix: See attached source code and event file.

2.2 Compilation/ directory tree:

```
├──.idea
│   └──inspectionProfiles
├──frontend
│   ├──public
│   ├──node_modules
│   └──src
│       ├──canvas
│       ├──components
│       │   ├──Backdrop
│       │   ├──Bookings
│       │   │   └──BookingList
│       │   ├──Events
│       │   │   └──EventList
│       │   ├──Footer
│       │   ├──Modal
│       │   ├──Navigation
│       │   ├──Spinner
│       │   └──Table
│       ├──context
```

```
│       └────pages
├────graphql
│    ├────resolvers
│    └────schema
├────helpers
├────middleware
├────node_modules
└────models
```

2.3 Deployment

Introduction

The entire project can run on a **stand-alone computer**. The application is running on port 3000.

Installation

If using project file without node_modules and dependencies, use npm install to install all dependencies.

Hence or otherwise, you can run the application by running npm start on the backend. (\main) and then running npm start on the frontend directory (\main and then running cd frontend terminal command)

View live

You can see the live website on the http://localhost:3000/ on any browser. E.g. Chrome and on any Operating system. E.g. Windows 10

Ensure that you are running on 100% zoom on a 1920x1080 display screen for optimal/ intended display as these are settings that the system was intended for.

Appendix

See attached project file for source code.

3. Traceability to design

I have used JavaScript/ NodeJS in the implementation for all components.

Design			Implementation
Class name	Package	Object name	Package
Maps API	Event	GoogleApiWrapper	Event
Email	Event	Nodemailer	App.js
Event	Event	Event	Event
Tester	Event	Authorization	Authorization
Time	Event	Cron	App.js
Planner	Event	Authorization component with planner resolver	Authorization
Consultant	Event	Authorization component with consultant resolver	Authorization
User	Accounting	Authorization component, with users resolver	Authorization
Tester	Accounting	Authorization component with testers resolver	Authorization
Graph	Reporting	CanvasJSChart	Dashboard Analytics
RevenueOverTime	Reporting	CanvasJSChart with options line	Dashboard Analytics
VerifiedTesterProportion	Reporting	CanvasJSChart with options pie	Dashboard Analytics
RevenueProportion	Reporting	CanvasJSChart with options pyramid	Dashboard Analytics
Booking	Booking	Booking	Booking

4. Testing

I will be unit testing the main use cases. These are the use cases which initially had a sequence diagram due to their importance.

Use case ID: 1	**Use case name:** Login
Test number: 1	
Objective: Test the primary path	
Set up: Email address ‘test@mail.com' and password ‘pass123’ is entered as login credentials and login form is submitted. The actor must be on the login page. The actor must not be logged in to an active session.	
Expected results: 1. The email and password entry is queried against entries in the database collection and a matching result is found 2. The entered password is hashed with 12 rounds of salting and is compared to the hashed password in the database. The passwords match. 3. A token is created for login context 4. Error message is not rendered 5. The user is redirected to the events page.	
Test: 1. Email address ‘test@mail.com' and password ‘pass123’ is entered as login credentials and submits form to log into an account 2. Ensure that email ‘test@mail.com’ and password ‘pass123’ entries are validated and passes test against length of password, presence check and format check for email. 3. Make sure token is created for login context. 4. Make sure that the there are no error messages about invalid user input 5. Ensure that the user is redirected to the events page.	
Test record	
Date: 18/06/2020	Tester: Neeraj Rai
Result: The correct email and password were entered but the system displayed an error message. A new token is not created. Found typo in salting function, which caused an incorrect round of salting.	
Date: 20/06/2020	Tester: Neeraj Rai
Result: Fixed. The user is redirected to the events page.	

Credit for table format: (Source: Lunn, Ken, Software Development with UML, Palgrave MacMillan, 2003, ISBN 0-333-98595-8)

Use case ID: 1	**Use case name:** DB connection - Login
Test number: 2	
Objective: Test the alternative path	
Set up: Email address ‘test@mail.com' and password ‘pass123’ is entered as login credentials and login form is submitted. Ensure that connection to Database is not available. Ensure that Use case ID 1 is unit tested and has passed.	
Expected results: 1. The backend receives the email and password parameters for login() from the GraphQL resolver as a model. 2. Users collection is queried to find the email and password in the database 3. Error message rendered on frontend due to no connection to database	
Test: 1. Enter Email address ‘test@mail.com' and password ‘pass123’ as login credentials 2. Submit login form 3. Ensure that error message is displayed regarding the loss of connection to database	
Test record:	
Date: 18/06/2020	Tester: Neeraj Rai
Result: Error message is not displayed. POST request error displayed on developer console.	
Date: 19/06/2020	Tester: Neeraj Rai
Result: Fixed. Error caught and message displayed on frontend to users.	

Credit for table format: (Source: Lunn, Ken, Software Development with UML, Palgrave MacMillan, 2003, ISBN 0-333-98595-8)

Use case ID: 1	**Use case name:** Invalid input login – login incorrect
Test number: 3	
Objective: Test the alternative path	
Set up: Email address ‘test@mail.com' and password ‘pass123’ is entered as login credentials and login form is submitted. Ensure that use case 1, test number 2 passes unit test The actor must be on the login page. The actor must not be logged in to an active session.	
Expected results: 1. The email ‘test@mail.com’ is queried against entries in the database collection and a matching result is found 2. The entered password ‘pass123’ is hashed and compared to entries in the database 3. Error message ‘Incorrect email or password entered message’ is rendered.	
Test: 1. Email address ‘aaaaaa@mail.com' and password ‘pass123’ is entered as login credentials and submits form to log into an account 2. Ensure that email ‘aaaaaa @mail.com’ and password ‘pass123’ entries are validated and passes test against length of password, presence check and format check for email.	

3. The entered password is hashed and compared to entries in the database. No user with email 'aaaaaa@mail.com' is found. 4. Ensure that 'Incorrect email or password entered message'
Test record:
Date: 18/06/2020 | Tester: Neeraj Rai
Result: Passes. The 'incorrect email or password entered' message is rendered upon incorrect credentials entry

Credit for table format: (Source: Lunn, Ken, Software Development with UML, Palgrave MacMillan, 2003, ISBN 0-333-98595-8)

Use case ID: 1	**Use case name:** Invalid input login – validation error
Test number: 3	
Objective: Test the alternative path	
Set up: Email address 'aaa' and password '123', ' ' are entered as login credentials and login form is submitted Ensure that use case 1, test number 2 passes unit test The actor must be on the login page. The actor must not be logged in to an active session.	
Expected results: 1. The email 'aaa', ' ' and password '123' entries are validated and they fail test against length of password, presence check and format check for email 3. Error message is rendered regarding the invalid login credentials	
Test: 1. Email address 'aaa' and password '123', ' ' are entered as login credentials and the login form is submitted 2. The email 'aaa', ' ' and password '123' entries are validated and they fail test against length of password, presence check and format check for email 3. Error message is rendered regarding the invalid login credentials	
Test record:	
Date: 18/06/2020	Tester: Neeraj Rai
Result: Error message was not rendered when inputting an email with invalid format 'aaa'. Database was queried with email 'aaa' and password '123', therefore failing (2).	
Date: 20/06/2020	Tester: Neeraj Rai
Result: Fixed. Email format validation implemented with length check for password.	

Credit for table format: (Source: Lunn, Ken, Software Development with UML, Palgrave MacMillan, 2003, ISBN 0-333-98595-8)

Use case ID: 11	**Use case name:** Create event
Test number: 1	
Objective: Test the primary path	
Set up: Planner must be logged in and is on the 'create a event' page. An existing event name must not be used while creating a new record. Enter event description 'Client x front page review', client name 'test', date '01/01/2020'. Consultant 'Bob test' and testers 'Test One', 'Test Two' are assigned to the event Session length '2' is assigned to the event Unit test 1, test number 1 must pass and none of its alternative flows should be triggered.	
Expected results: 1. A new event record must be recorded on the database 2. Consultants and testers assigned to the event 3. Event description and client information assigned to the event 4. Session length is assigned to the event 5. Event and testers are dynamically linked with their correct cardinality 6. Email is sent to the planner and a reminder email is sent to testers a week before testing. 7. Booking record is created from event with the timestamps.	
Test: 1. Enter event description 'Client x front page review', client name 'test', date '01/01/2020'. 2. Consultant 'Bob test' and testers 'Test One', 'Test Two' are assigned to the event 3. Session length '2' is assigned to the event 4. Event and testers are dynamically linked with their correct cardinality 5. Email 'You have booked a new event' is sent to planner and email 'You will have a testing session at "location name" at "time" with "consultant name" is sent to testers.	
Test record:	
Date: 18/06/2020	Tester: Neeraj Rai
Result: Passed	

Credit for table format: (Source: Lunn, Ken, Software Development with UML, Palgrave MacMillan, 2003, ISBN 0-333-98595-8)

Use case ID: 11	**Use case name:** Cancel
Test number: 2	
Objective: Test the alternative path	
Set up: Planner must be logged in and is on the 'create a event' page. Enter event description 'Client x front page review', client name 'test', date '01/01/2020'. Unit test 1, test number 1 must pass and none of its alternative flows should be triggered.	
Expected results: 1. All inputted event details such as event description 'Client x front page review' is cleared. 2. The planner is redirected to the root page/ bookings page.	
Test: 1. Enter event description 'Client x front page review', client name 'test', date '01/01/2020'. 2. Press cancel button 3. Ensure that the page is redirected to root page/ bookings page. 4. Ensure that previously inputted details are cleared by going to the 'Create a event' page.	

Test record:	
Date: 18/06/2020	Tester: Neeraj Rai
Result: Passed	

Credit for table format: (Source: Lunn, Ken, Software Development with UML, Palgrave MacMillan, 2003, ISBN 0-333-98595-8)

Use case ID: 11	**Use case name:** Session time out
Test number: 3	
Objective: Test the alternative path	
Set up: Planner must be logged in and is on the 'create a event' page. Enter event description 'Client x front page review', client name 'test', date '01/01/2020' Unit test 1, test number 1 must pass and none of its alternative flows should be triggered.	
Expected results: 1. Planner has been on the page exceeding an hour 2. The session token expires 3. The user is redirected to the login page	
Test: 1. Enter event description 'Client x front page review', client name 'test', date '01/01/2020' (optional) 2. Have page open for 1 hour. Token expiration value can be changed to 10 minutes, '10m' during testing 3. Ensure that the page is redirected to root page/ bookings page after the time on the page has exceed the token expiration time. E.g. After 1 hour.	
Test record:	
Date: 18/06/2020	Tester: Neeraj Rai
Result: Passed	

Credit for table format: (Source: Lunn, Ken, Software Development with UML, Palgrave MacMillan, 2003, ISBN 0-333-98595-8)

Use case ID: 11	**Use case name:** Email exception
Test number: 4	
Objective: Test the alternative path	
Set up: Planner must be logged in and is on the 'create a event page. Email 'You have booked a new event' and email 'You will have a testing session at "location name" at "time" with "consultant name" is is set up as template email to send. Unit test 1, test number 1 and 11 must pass and none of its alternative flows should be triggered.	
Expected results: 1. Planner has booked an event 2. Email does not send to users 3. Error message is displayed	

Test: 1. Planner has booked an event. 2. Email does not send to users 3. Ensure that error message is displayed	
Test record:	
Date: 18/06/2020	Tester: Neeraj Rai
Result: Error message does not show.	
Date: 22/06/2020	Tester: Neeraj Rai
Result: Open. The issue is still awaiting a fix.	

Credit for table format: (Source: Lunn, Ken, Software Development with UML, Palgrave MacMillan, 2003, ISBN 0-333-98595-8)

Use case ID: 11	**Use case name:** DB connection – Create event
Test number: 5	
Objective: Test the alternative path	
Set up: Planner must be logged in and is on the 'create a event' page. An existing event name must not be used while creating a new record. Enter event description 'Client x front page review', client name 'test', date '01/01/2020'. Consultant 'Bob test' and testers 'Test One', 'Test Two' are assigned to the event Session length '2' is assigned to the event Unit test 1, test number 1 must pass and none of its alternative flows should be triggered.	
Expected results: 1. User attempts to create a new event. 2. A MongoDB event model is generated from user input. E.g. clientName : 'Client name' 3. The new model is inserted to the DB collection for events 4. Error is displayed as there is no connection to DB	
Test: 1. Enter event description 'Client x front page review', client name 'test', date '01/01/2020'. 2. Consultant 'Bob test' and testers 'Test One', 'Test Two' are assigned to the event 3. Session length '2' is assigned to the event 4. A MongoDB event model is generated from user input. E.g. clientName : 'Client name' 5. The new model is inserted to the DB collection for events 6. Error is displayed as there is no connection to DB	
Test record:	
Date: 18/06/2020	Tester: Neeraj Rai
Result: Passed	

Credit for table format: (Source: Lunn, Ken, Software Development with UML, Palgrave MacMillan, 2003, ISBN 0-333-98595-8)

Use case ID: 11	**Use case name:** Maps exception
Test number: 6	
Objective: Test the alternative path	
Set up: Planner must be logged in and is on the ‘create a event page. Planner should only be able to select testers with to be view in maps view, which renders nearby users as pinned locations. Unit test 1, test number 1 and 11 must pass and none of its alternative flows should be triggered.	
Expected results: 1. Maps will render a text message that map cannot be rendered. 2. Another error message will be rendered to let the planner know that they should try using the system again later.	
Test: 1. Planner must be currently booking an event 2. Planner selects multiple testers to view their nearby location together on the map 3. Error message is displayed that the request cannot be resolved due to third party errors.	
Test record:	
Date: 18/06/2020	Tester: Neeraj Rai
Result: Error message does not show.	
Date: 24/06/2020	Tester: Neeraj Rai
Result: Fixed. Error message is displayed as error is caught from the API.	

Credit for table format: (Source: Lunn, Ken, Software Development with UML, Palgrave MacMillan, 2003, ISBN 0-333-98595-8)

Use case ID: 35	**Use case name:** Display Analytics
Test number: 1	
Objective: Test the main path	
Set up: Planner must be logged in and analytics dashboard page Tester records and events must exist in the database which will used to render graphs. **Sample tester:** Disability:Array Devices:Array testingLocation:Array Assistance:Array firstName:"Cynthia" lastName:"Patrick" addressLineOne:"58 Nobel Freeway" town:"Voluptate exercitati" country:"United Kingdom" postcode:"Hic modi nobis eos " email:"bybeqecu@mailinator.com" telephoneNo:"+1 (347) 671-6472"	

specialHardware:"Non praesentium iste"
additionalInformation:"Facilis ex laboris l"
standby:false
hearAbout:“Our website"
dob:"1973-07-23"
techSkill:"beginner"
rating:0
status:true

Sample events:
_id:
5ef60b80625675528c3a4468
title:"Client Testing/ 1 Tester"
description:"Testing description"
price:2000
date:2020-04-03T05:20:00.000+00:00
creator:5ef5fb51625675528c3a4466
adrNumber:"1"
clientName:"Name"
projectDescription:"Review for front page"
sessionLength:"2"
testers:"Cynthia Patrick"
consultants:5ef5fb51625675528c3a4466
__v:0

Unit test 1, test number 1 and 11 must pass and none of its alternative flows should be triggered.

Expected results: [See setup for tester and events input]
1. The data from tester and events models are correctly parsed by canvas JS
2. Graph about proportion of unverified to verified testers are displayed
3. Graph about revenue overtime is displayed
4. Graph about pyramid of profit to sales is displayed

Test:
1. The data from tester and events models are correctly parsed by canvas JS
2. Graph about proportion of unverified to verified testers are displayed
3. Graph about revenue overtime is displayed
4. Graph about pyramid of profit to sales is displayed

Test record:

Date: 18/06/2020	Tester: Neeraj Rai

Result: Passes

Credit for table format: (Source: Lunn, Ken, Software Development with UML, Palgrave MacMillan, 2003, ISBN 0-333-98595-8)

Use case ID: 35	**Use case name:** DB connection – Display analytics
Test number: 2	
Objective: Test the alternative path	

Set up: Planner must be logged in and analytics dashboard page

Tester records and events must exist in the database which will used to render graphs.

Sample tester:

Disability:Array
Devices:Array
testingLocation:Array
Assistance:Array
firstName:"Cynthia"
lastName:"Patrick"
addressLineOne:"58 Nobel Freeway"
town:"Voluptate exercitati"
country:"United Kingdom"
postcode:"Hic modi nobis eos "
email:"bybeqecu@mailinator.com"
telephoneNo:"+1 (347) 671-6472"
specialHardware:"Non praesentium iste"
additionalInformation:"Facilis ex laboris l"
standby:false
hearAbout:“Our website"
dob:"1973-07-23"
techSkill:"beginner"
rating:0
status:true

Sample events:
_id:
5ef60b80625675528c3a4468
title:"Client Testing/ 1 Tester"
description:"Testing description"
price:2000
date:2020-04-03T05:20:00.000+00:00
creator:5ef5fb51625675528c3a4466
adrNumber:"1"
clientName:"Name"
projectDescription:"Review for front page"
sessionLength:"2"
testers:"Cynthia Patrick"
consultants:5ef5fb51625675528c3a4466
__v:0

Unit test 1, test number 1 and 11 must pass and none of its alternative flows should be triggered.

Expected results:
1. The fetched records are not processed as there is no database connection
2. Error message that there is no connection to database is rendered

Test:
1. As analytics dashboard component is mounted via react router, the records for Tester and Events are fetched.
2. The request does not pass as there is no connection
3. Error message that there is no connection to database is rendered

Test record:	
Date: 18/06/2020	Tester: Neeraj Rai
Result: Passed	

Credit for table format: (Source: Lunn, Ken, Software Development with UML, Palgrave MacMillan, 2003, ISBN 0-333-98595-8)

5. System testing

For any given month, I believe that there will be 1,000 users using the system. This is an estimation to the number of users as there are currently no analytics on the number of users who actively visit the AbilityNet website. There may be many more or around the estimated value for the number of users visiting the application page to become a tester. The main functionality of this web application resolves around managing the data and booking events. Only AbilityNet internal employees will be able to access this information, hence I estimate that there will be about 10-20 people using the system's core features, few times a month.

'Apply to be tester' will therefore be the most important use case. I assume that the use case will have a probability of 0.2 if the application form is on the AbilityNet website. This is because the website has a lot of other functionalities and features, so much of the traffic may not be interested in becoming a tester. However, as a proportion of the importance of this use cases to other User Testing Process use cases, it will have a higher system usage probability of > 0.8 (E.g. 0.98) as it is the part of the system with highest traffic. It will be therefore important to test the input forms. E.g. Use case Create tester record, which may have a probability of 0.4. Assuming 0.4 of those 0.8 users will want to become a tester for AbilityNet, give that the use-case is **randomly chosen**.

As a proportion of the 0.02 AbilityNet employees who are using the internal features:

From the 0.02 (1-0.98) of AbilityNet employees (20 out of 1000) who will not be using the apply tester form, 0.5 of those users (0.01) from the whole system will be using the Create events feature, as it is the main feature of the system. This includes all the dependent use cases. E.g. Finding testers or adding event description while creating an event. 0.1 of such users may be using the update email use case and 0.2 may be using the analytics dashboard at any time. 0.1 of consultants logged in will be using the system to verify user and give ratings.
The probability of the use-case N will be 1-(0.2 + 0.1 + 0.5 + 0.1) of the 0.02 proportion of AbilityNet employees.

Simulating the testing

A simulated automated testing of N representative tests will be a loop of N interactions. For the probability distribution, a random number generator can be used to pick use case randomly to simulate the distribution of use case usage, rather than using N amounts of each use case.

For use cases with different parameters, I will be populating them using dedicated RNG values which follow a pattern and range.

E.g. For use case 2 – Login:

I will be using a:

- parameter for username, password which will be string of up to 20 characters. Users can also use email instead of username, so a randomly generated string followed by "@{random string or dictionary of mail providers} . {random string or email supported domains} to be supplied as a email inputs.

E.g. For use case 3 – Create event:

I will be using a:

- parameter for price which will be an integer in the range [0…999].

Additionally, different testers will have to be selected. This mechanism is more complex than inputting a value as a tester will have to be added into a project by clicking an entry on a table. To do this, an internal to automatically select testers during testing or external macro script could be written to target a position on the screen. The selection should be random and different testers should be picked each time.

The values for these parameters can be generated at random using a dedicated RNG that will have to be adjusted to return the type and in the needed range. E.g. as mentioned above, the integer price parameter [0…1] could be multiplied by 100 and converted to an integer so it comes in the range [0…999].

Judging and recording

I will be recording the outcome of each use case using a judging mechanism such as an oracle such as **Jest JS.** I could use it to generate text and validate input or check the post condition. E.g. check database for any signs which show that there was a successful outcome of a test.

With Jest I can automate the generation of parameters to input into testing E.g. Strings.
Const text = generateText('Name', 'Password')

Augmented test conditions can be written as following:

```
Exports.checkAndGenerate(name, password) => {
    Return (!inputLogin(name, true, false) || !inputlogin(password, false, true)) ? false : true
}
```

If I want to test the summation of two numbers for calculating total revenue from sales for generating the graph, I can use

```
Test('adds ${val1} + $}{val2} to equal ${valSum}', () => excpect(sum(val1, val2)).toBe(val1+val2));
```

I will then record the results from the test into a database.

6. Nonfunctional testing

Security:
I can test for security and fix weaknesses by hiring cyber security firms to find any issues with the system by using brute force methods such as DDOS or penetration testing.

Scalability:
I can test for scalability by creating a load of 10x more users than in normal operation and see how the response time of the web application is affected. This will ensure that the applications runs optimally during web traffic and continues to do so when there is a larger traffic.

Reliability & availability:
I will use the system testing infrastructure to simulate a year of integration and based on the results I have stored, I will see how many failures there are any it will show the value of reliability and the % downtime of the system experience based on the failures. I will look at any patterns in downtime from the data and attempt to trace any reliability or availability problems and try to remedy it.