Software Engineering Group Project

COMP2002 / G52GRP: Interim Report

Project information:

Project title: BrowneJacobson-SelfService

Project sponsor: Browne Jacobson

Academic supervisor: Mercedes Torres Torres

Industry Sponsor: (if applicable)

Industry Supervisor(s): Steve Brooks

Team information:

Team number: 12

(Callum, Davies, 20191417, psycd6)

(Thomas, Murphy, 20158976, psytm1)

(Zixiang, Jin, 20126692, scyzj3)

(Shuxiang, Hu, 20124816, shysh1)

(Zihui, Xu,20164269, efyzx1)

(Ashley, Nnawugo, 20171111, psyan6)

Documentation (links):

Trello board:

https://trello.com/nottinghamteam12

Code repository:

https://projects.cs.nott.ac.uk/COMP2002/2020-2021/team12_project/tree/master/src

Document repository:

https://projects.cs.nott.ac.uk/COMP2002/2020-2021/team12_project/tree/master/docs

Further Online Documentation (if applicable):

Table of Contents

Initial analysis

Project brief

A big law firm like Browne Jacobson can have many employees traveling around. However, planning a trip could be frustrating, for example, booking trains and hotels on different websites through their mobiles, searching for confirmation emails received months ago, etc. An unpleasant travel experience could jeopardize the quality of employees' services. Hence, an electronic traveling management method is of essential importance to ensure the staff of Brown Jacobson a smooth and pleasant trips.

Beyond the brief

There are still aspects worth considering but were not mentioned in the brief and pitch.

Considering should be given to what data should be kept in the database. After discussion, all traveling related data will be stored in a database, including start point, destination, travel mode (taxi, driving, train, etc.), start date, and end date. According to the industrial supervisor, Steve Brooks, no extra effort is needed to the storage of user account since details of the law firm's employees are already kept in their database. This means users don't have to register before using the APP as well.

Another extra non-functional requirement we identified was performance, how to make sure the APP responds quickly. Since a database will be used to store data related to user's traveling and the total amount of data can grow quickly when the APP is put into use, the usage of SQL queries can have an extraordinary effect on performance. Hence, all the team members are required to self-study SQL performance tuning and query optimization during the requirement and specification phase so that no time will be wasted on learning these methodologies when the APP is connected to a database.

The intention of designing such an APP is to simplify the traveling of employees within the law firm rather than to earn a profit. Hence it should be tailored for staffs from Brown Jacobson. To achieve that, one approach is to remove the register procedure as discussed before. Besides, we decide to use a color scheme similar to that of the company's official website, so that the users may feel proud knowing that they are using a good APP provided by their own company.

Apart from the functional requirements discussed in the brief, we also decide to add an intelligent route planner so that users will not get lost in a strange city. Otherwise, massive time will be wasted on find routes and the business may be ruined. After meetings with industrial supervisor, user stories derived from this requirement were assigned high priority not only because it is useful in real-life situation, but also it paves the foundation for many other functionalities.

Existing products

To identify what will be the motivation for employees to use our APP as an alternative for existing powerful products, we conducted a detailed investigation into similar mobile applications.

	Google map	Uber	TripIt	Hopper	Didi	Meituan
Main functionality	Routing	Taxi booking	Traveling management	Transportation/Accommodation booking	Taxi booking	Hotel booking
Intelligent Routing/Recommendation	Yes	Yes	No	No	Yes	Yes
Traveling reservation	No	Yes	No	Yes	Yes	Yes
Travel management	No	Yes	Yes	Yes	Yes	Yes
Hotel reservation	No	No	No	Yes	No	Yes
Parking reservation	No	No	No	No	No	No
Accessible via any platform	Yes	Yes	Yes	Yes	Yes	Yes

Area	Global	Over 10,000 cities	Global	Global	Global	China

Table 1. Existing product analysis

According to table 1, none of the existing products can offer sufficient service as required in the project brief. Hence, the biggest advantage of Self-Service over similar APPs might be a one-stop traveling service system. Therefore, the main purpose of this project is to offer a one-stop, service-integrated mobile app that allows users to book trains, hotels, and related items, view future bookings all in one go.

Requirement Analysis

Requirement Prioritization

Since we are using Agile development with sprints, the requirements are documented as user stories, which can be put into each sprint. By having regular weekly meetings with our sponsor, Browne Jacobson LTD, we have confirmed 30 user stories so far and each one of them was assigned with an effort rating (estimated time cost) and a priority level (what to do first). Effort ratings are 1-10 integers given based on our consensus on the various workloads of each user story (a "1" takes the least amount of time where a "10" takes the most). We currently have four priority levels: High, Medium, Low and Lowest. User stories are categorized into those levels based on the following rules (see Appendix A):

1. High Priority User Story ("Must-Have"): the program needs the functionality and will not be considered as a complete functioning program without it.
2. Medium Priority User Story ("Should-Have"): the program would be able to function without this functionality, but it would be very useful to have it. They are still important but not as critical as those of the high priority levels.
3. Low Priority User Story ("Could-Have"): it will improve the performance or user experience of the program, but it is not necessary.
4. Lowest Priority User Story ("Won't-Have"): the current version of the program does not require this functionality.

Before each sprint, we pick out the ones that we would like to put into the next sprint, according to their priority levels, from high priority to the lowest. Then we decide how many of those user stories will be put in the next sprint based on their effort ratings. For example, we can fit two "5" user stories or a "10" user story into one sprint. Those user stories will then be broken down into smaller, specific user stories which can be assigned to each member of the team.

Requirement Validation

Most of the user stories we came up with so far are confirmed and accepted by our sponsor, however, two of them (User story 29 and 30) are categorized as the lowest priority level (Won't-Have) after we presented them to our sponsor as they are not in the scope of the expected functionalities of this program, i.e., employees don't need to be able to book flights or travel overseas. Therefore, we reduced our scope of only domestic land transportations (train and taxi), which avoided any waste of time and effort. Other than that, we have not encountered any changes to the requirements so far, however, we are aware of the benefits of acknowledging any changes to the requirement as soon as possible. Therefore, we will keep meeting with our sponsor regularly and react to any possible changes to the requirement and update our user stories.

Due to the persistent pandemic, it might not be practical for our team to do focus groups and field observations. Our current requirement gathering method is to have weekly meetings with our sponsor. However, we will try to explore more possible approaches as we go along the project, such as online surveys and interviews with the employees of Brown Jacobson to get a broader view of the requirements of the application.

Development tools

Framework and Language

Flutter, an open-source, cross-platform mobile app software development framework by Google which is written in the Dart language, is used in this project. It is powerful in generating codes could run on both iOS and Android platform, perfect for end-user having varying types of phone. Google said:" Flutter can help programmers develop mobile applications with beautiful interfaces more easily and quickly." Dart, a network programming language developed by Google, helps developers build browser oriented high-performance applications efficiently.

Dart allows targeting desktop and mobile browsers, dart applications to run in dart virtual machines, or be compiled to run as JavaScript. Dart VM is much faster than the JavaScript V8 engine used by chrome. For developers who have experience with Java or JavaScript, flutter is easy to learn and use, which is the case for team 12.

Hardware

As for hardware requirements, Flutter needs Operating System: Windows 7 SP1 or later (64-bit), x86-64 based. Disk Space: 1.32 GB (does not include disk space for IDE/tools).

Tools: Flutter depends on these tools being available in your environment.
*Windows PowerShell 5.0 or newer (this is pre-installed with Windows 10).
*Git for Windows 2.x with the Use Git from Windows Command Prompt option.

If Git for Windows is already installed, make sure you can run git commands from the command prompt or PowerShell.

For the APP, no hardware is needed so far since Android Studio and XCode offers useful virtue machine for testing.

Miscellaneous tools

We are using Flutter SDK as our main library, which is bundled with Dart SDK.
The main IDE is Android Studio which is based on IntelliJ. In plugins of Android Studio, we employed Flutter and Dart plugins for the flow of flutter developers (running, debugging, hot reloading, etc) and code analysis (code validation while typing, code completion, etc).
In terms of mobile APP emulator, Android Studio and XCode support Android and IOS emulator respectively. The latter has an advantage over Android Studio when emulating in causing less CPU load than. We also used UI designing app, for example, Proto.io. Common and powerful version controls tools including Git and Gitlab are introduced for project management.

Team Management

General Management

As an organized group we usually chat on WhatsApp and hold meetings on Microsoft Teams. The meetings would usually take place on a Monday afternoon, a Wednesday morning to the afternoon and on a Friday morning or afternoon. The meetings at the start involved how to install Flutter, how to make it run (getting emulators and simulators working) and making sure everyone has the same starting SRC file, this was done by creating an SRC file which was pushed to Git and pulled by the whole team. Later meetings involved dividing the workload, fixing issues in the code as a group and planning the future steps of development.

Sprint workflow

We organize the development workflow into two-week long sprints. In the beginning, user stories to be achieved in the sprint will be identified according to priority and estimated effort.

Then, for each week we usually hold three meetings, on Monday, Wednesday, and Friday respectively. The meetings on Monday are usually for workload allocation and coding. During Wednesday's lab work time, we would have discussions to solve technical difficulties and fix tough bugs. One member of the team would share their screen and work on the code whilst one or two members would observe, point out era's and give suggestions of ways to move forward on the task. In meetings on Friday, there will be demonstrations of weekly achievements to Steve and discussions on future plans.

At the end of a sprint, there will be a retrospective section with Steve. We would list "what went well", which is usually our achievements, "what didn't go well" that we had been struggling with and "what still puzzles us" to highlight the areas that we are still confused about. This was very useful as it would allow us to plan how to more effectively tackle future sprints. It also helped our sponsor to have significant insight into what we have been doing and what exactly we are struggling with, so he could offer his advice.

Also, to make the most of the meetings we made sure that all details of the meetings were noted down. This was beneficial, as we were able to record what had been discussed in the meetings especially for those who may not have been present can catch up using these notes. The notes were also beneficial for us to refer to whilst we are working and have a clear goal of what needs to be implemented from those meetings.

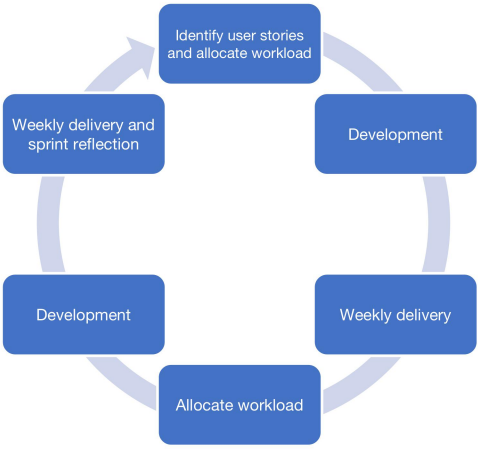

Figure 1. Sprint workflow

Management Tool

We used Trello as our management tool, first and foremost we had a board which we named "User Story Backlog", this board contained all the User stories which we had written up as a team concerning how the Self-Service app would be used by real-life people. The backlog contains all the user stories, the user stories are then further broken down into smaller ones and then into what needs to be implemented and coded for it to work. The user stories remain under the section of the board called "To be allocated', the broken-down stories are then under the section of the board called "Allocated", in the "Allocated" section where the broken-down user stories were assigned to the team members by our Team Leader. The next section on the Trello sprint board is "completed", this section is for all the completed broken-down user stories.

The Trello board works by setting up lists from left to right, we set it up by going in the order of "To be Allocated" to "Allocated & Working on" to "Completed". To each list, you can add cards which you can add what the task involves, who the task is going to be done by, when the task is set and when the due date of the task.

Each card can be moved from one list to another once it reaches the criteria of that specific list, for example, it can only be in the "Allocated & Working on" list when the task has been assigned to an individual and they can start working on it. The Trello board helps us as a team know who is working on what specific task, how many tasks we all as a collective must complete when the tasks must be completed.

Initial Software Implementation & Testing

Requirements and Specification

After requirements gathering and validation with Steve, requirements are translated into user stories and further broken down into atomic, measurable tasks. We narratively designed a model of the end-user using the app in a simple step-by-step process. This took the form of a nested list of operations, substituting a sequence diagram from the sole perspective of the user. For example, the app would offer the user hotel options in London, if they selected that they would like to plan a journey to London on the previous page.

- Homepage
  - Button to taxi reservation
  - Button to train reservation
  - Button to parking reservation
  - Button to hotel reservation
  - Button to view history reservations
  - Button to log out

Figure 2. Segment of Prototype Outline (Our substitution for a sequence diagram)

This turned out to be useful because our team is now aware of what data needs to be shared between each page (class) in the application.
A request from our client, Steve Brooks, has been to maintain the security of Browne Jacobson's staff. As a result, we have agreed to use a dummy data scheme. We aim to develop the app for a general user and if time allows a connection to Browne Jacobson's internal authentication system, where user information is protected.

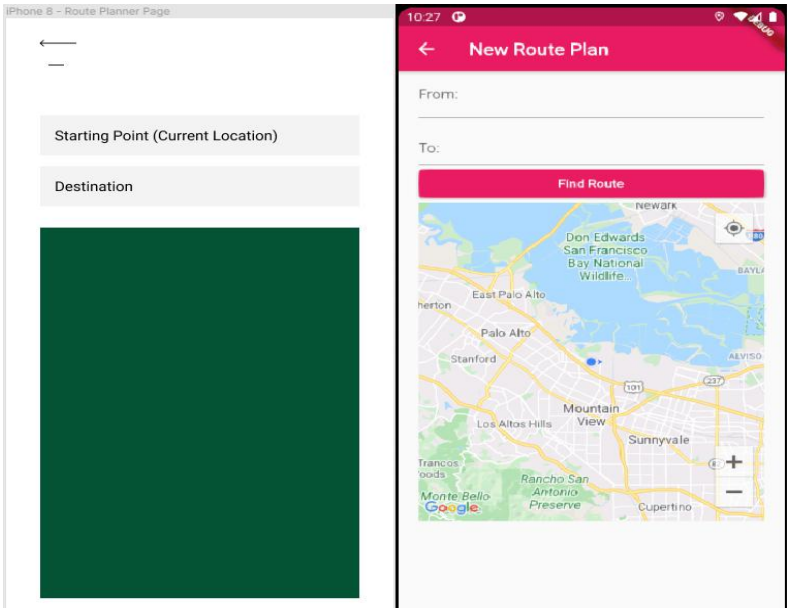

Figure 3. Initial low-fidelity prototype made in Figma (Left) vs Current implementation (Right)

Our application serves to be a developing prototype. Starting from a series of low-fidelity diagrams of how pages would be structured and how buttons would function, we've developed a foundation of the mobile app, demonstrating the page structure and intended colour scheme to match the end-user company's aesthetics.

This can be further improved in later sprints by communicating discoveries quicker. For example, we designed the app to have a banner that would allow the user to move back. However, we later discovered the app bar, which was more user-friendly, offering a simple return option. From here, our design evolved, allowing the homepage to appear as a hub, providing large UI elements for page navigation that is requested by a non-functional user story.

Implementation with Version Control

We've utilized Git's branching and merging tools to allow concurrent development. These have been very useful for maintaining the integrity of the system, preventing common concurrency issues such as overwriting.

Our strategy for conflicts, which have occurred, has been to gather at the end of every half of a sprint to merge the branches with all team members present. Here we can discuss conflicts, through peer programming and decide between us which segment of code to overwrite, and where we can migrate the overwritten code if necessary. We discovered that the cause of our first conflict was the fact that two different frameworks were used for the body of the app's homepage. As a result of this, we aim to communicate user stories that affect the app globally, such as a new page or a new piece of information to trace between pages.

Specifically, each of our team members forms a branch for each user story they would like to implement, following a camelCase naming scheme. This is advantageous because it allows Thomas as the Git manager to consult the Trello board to identify what branch somebody is working on if they have a problem.

If there is a case where a user story depends on functionality added by another, such as an input box for the taxi page, while another member is tasked with creating a taxi page, we communicate in meetings on who is most likely to be able to work on their user story sooner, allowing us to allocate the tasks chronologically. In circumstances where the roles have been reversed, we've swapped the tasks between the members to maintain a good development pace.

In the event of a user story that is dependent on another being completed first, it may be branched onto the master if manual testing under a controlled environment allows it. This is if the team member can ensure their segment works by testing it on a stable page that provides the same interface, such as moving the taxi input box to the app's home page temporarily. This allows us to keep the master branch up to date with functional code.

If a user story's functionality depends on another too much for it to function without it, the aim is to merge it onto its dependent branch first. This decomposes the task of merging all the branches in a manageable way that doesn't result in conflicts and re-writing of the same code.

An example of this can be seen in the diagram where "phoneLocationAccess" is branched with "DestinationInput" as a result of monitorability of the success of "phoneLocationAccess" depends on the field implemented by "DestinationInput".

Approaches to Testing

Black-box testing was utilized, following a combination of both atomic and non-atomic stories. To test a user story, we would run the app and interact with the new feature. For example, we experimented with typing different values into the location box, including empty, reasonable and erroneous field options. After confirming the expected outcome of the new feature, our team members would follow the procedure of navigating through the app, raising any concern if another segment of the program broke with a comment in a meeting which can be added to the Trello board as a bug, as an alternative to Git's issue board.

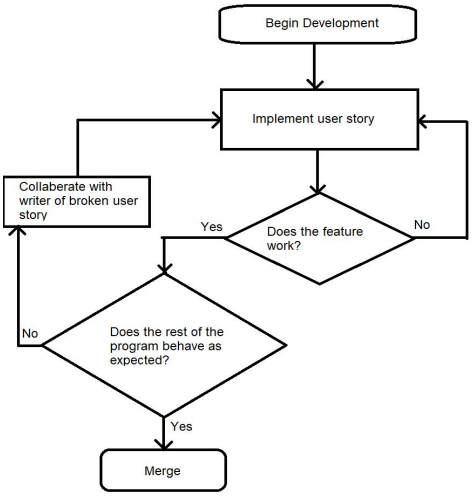

Figure 4. Flowchart of testing and maintenance

Bugs were tracked in Trello as a result of some of the problems that arose during Sprint #2. A temperamental bug exists where an error is briefly displayed in some testing environments, as a result of asynchronous data initialization and usage, masked by abstracted multi-threading and concurrency in the program. As a result, we have made it a priority to implement bug maintenance as priority user stories in our sprint Trello boards.

Our team members are new to programming for mobile apps, making the implementation of unit testing a learning process. To maintain the integrity of the app and serve as a monitor of where a branch is safe to merge onto the master, we have set up some core unit tests such as navigation between pages. This is advantageous because one can quickly test if their segment code affects external segments of code to their current target. We wish to develop this to include more specific interactions as the program grows and evolves from manual testing.

The area we would like to improve on for testing is how we document manual testing. Currently, team members have used their own intuition, considering the design for the working app and testing if the outcome of the test matches the expected outcome.

Expanding on referencing our designs, we believe manual testing will become more monitorable if we begin implementing a formal written test plan which members of the team can refer to, when confirming to merge a branch.

Reflection

In the time that has passed since our group was matched with a project, we have completed 2 biweekly sprints, totaling 4 weeks of development work on a prototype. Upon reflection, there are many lessons to be learned from this period that can be used in the future to improve our group's productivity and workflow.

Difficulties

The first point worth reflecting on is our difficulty with learning the programming language flutter. In our case, we had a maximum of one week to learn how to use this language before using it to develop a prototype for an app. This has proved to be an insufficient amount of time to learn a new language. While we were able to use the language and develop a working prototype, our lack of knowledge caused us to face bugs which would result in delays. As a consequence of these initial issues, we learnt that one of the best ways to resolve bugs is to use peer programming.

Peer programming is an extremely valuable technique when facing difficult bugs. During our first sprint, each member encountered bugs that they found difficult to fix alone. The peer programming technique allowed us to tackle these issues together, using our shared experience of the language to overcome the issues that we faced individually. This experience taught us that teamwork is extremely valuable and that it is the best way to eliminate overheads when producing deliverables. Moving forward, this technique is the method we will use when resolving issues with difficult bugs.

An issue we faced that caused some delay in our progress was accessing and using the Google Maps API. This issue was encountered during the second week of the first sprint, which was when we planned to use the maps API in our prototype.

To get access to the API, we needed to provide payment details. The team expected to have these details provided by The University/Browne Jacobson. However, neither could provide the payment details we needed. This issue was resolved by using the personal payment details of a member of the group; however, this took a toll on the progress we could make. In reflection, this could have been avoided, had we known from the start what the university's policies were on providing payment details.

Achievement

To date, we have developed a low fidelity prototype of a mobile application. This prototype demonstrates a very basic user interface where the user can access various pages that will later be filled with functionality. The home page contains buttons to three other pages: The route planner page, The book train page, The book taxi page. The route planner page contains the most functionality: On this page, the user can enter their location and destination, and see a route marked out on a map from point to point. The book train and book taxi pages have

more limited functionality: A user can input their locations and destinations, and these inputs are validated by a method to check if the inputs are empty.

What can be improved

Reflecting upon project management, there are a few things that can be said. During both sprints, we used Trello to keep track of user stories and the user story backlog. While our usage of the Kanban sufficed for our needs, it was not utilized for maximum effectiveness. For instance, user stories on a sprint board did not reflect on the user story backlog board; meaning two boards had to be maintained for each user story. It would be well worth getting to understand Trello better so that tracking user stories is more efficient.

Another reflection worth commenting on is that team meetings should have more structure. During our sprints, our approach to meetings was to discuss what user stories we were going to attempt during the meeting, and delegate who was responsible for each story. Having a stricter process of completing user stories will ensure that all resources are employed and that the group achieves maximum productivity. Moving forward, a better approach would look like this:

1. Identify prerequisite stories/order of stories to be completed
2. Delegate user stories to group members
3. Set deadline for user stories
4. Confirm visual design of each user story (if applicable)
5. Have interim discussions about user story progress
6. Employ peer programming if no progress made
7. Reassess user story progress at end of meetings

Individual contribution

Each member of the team has been given equal workload and opportunity to contribute to the project. Thomas has used his knowledge of git to manage a remote git repository and also created the prototypes framework. Shuxiang has actively managed the group's administrative work and added functionality by using the maps API to get routes from one location to another. Kieran, Ashley and Zihui have contributed by creating the input functionality and validation seen throughout the prototype. Callum contributed to the team by managing the team's direction and adding functionality to access the user's location.

Summary & Future Plan

Both industrial supervisor and the whole team agree that team 12 has made satisfying progress this semester, including gathering and validating requirements, learning to work with Flutter, integrating Google Map APIs and delivering a prototype, which are supported by good team management skills.

In the development phase of the next semester, we will be focusing on the following aspects:

- Integrate all the required APIs into the software
- Improve UX (smooth navigation between pages, user-friendly prompts, quick response etc)
- Connect the software to a database.
- Add more automated unit testing

Appendix A

User story lists

User Story ID	As a <type of user>	I want to <perform some tasks>	So that I can <achieve some goals>	Priority
1	employee who will be on a business trip	know the available transportation and make a reservation	plan my journeys	High
2	employee who will be on a business trip	search for available hotels and make a	Ensure my accommodations	High

		reservation.		
3	employee who drives a car to meetings	search for available parking spaces near my destination	ensure I have a place to park or to take alternative transport	High
4	employee who travels a lot	have a record of my past traveling and reservation receipts	see all the bookings I have made	High
5	employee	login to the app	access my personalized page and book work related items	High
6	employee	open a navigation menu	navigate to different pages within the app	High
7	employee who travels around the country	have a route planner	get from A to B in the shortest amount of time	High
8	employee who is indecisive and clumsy	amend my bookings at any point before the day of my journey	my mistakes can be fixed	High
9	employee who travels a lot	make multiple reservations on transportation at the same time	avoid waiting for one travel to be finished before planning another, which might be too late	High
10	employee who is organized	have multiple journeys in my planner	can perform all my preparations in bulk	High
11	employee whose business trip was canceled	be able to undo my reservations on transportation, accommodation and parking	my trips can be arranged in a flexible way	High
12	employee who doesn't know the locale of the other cities very well	have recommendations on travel options	ensure I choose the most efficient and comfortable travel so that during a trip, I know where I am going, and no time is wasted due to getting lost	Medium
13	employee who has an old phone	have a large UI elements option	can use the planning app with ease	Medium
14	employee who doesn't know other cities very well	be informed about hotels in the areas	can find a place to stay during visits	Medium
15	employee who travels a lot	make multiple reservations on accommodation at the same time	be prepared in advance	Medium
16	employee who travels a lot	make multiple reservations on transportation	can be confident that I will get to my destination on time	Medium
17	employee	open a settings menu	personalize my experience in the app and edit my personal details	Medium

18	employee who doesn't use modern interfaces	have a walkthrough guide of the app	use a planner app to help me plan my journey	Medium
19	employee who struggles planning transactions	have a system to automatically book my expenses	I don’t need to do it myself	Medium
20	employee who is anxious about changes	receive immediate notification if there is a problem or cancellation with my planned journey	I can re-plan my travel	Medium
21	employer	forward my employees their destination for a meeting	ensure they know where they are going	Low
22	employee who is clumsy	have the ability to retrieve my information on a new device	keep my bookings in a situation where I lose my phone	Low
23	employee with short attention	have graphical user elements as an alternative	read large passages of text and fill out written forms	Low
24	employee who is anxious about changes	reserve back-up bookings	I can still travel if cancellations occur	Low
25	employee who is a perfectionist	receive updates if better opportunities for hotel bookings arise before the day of my journey	avoid wasting time	Low
26	manager who oversees the work my subordinates	have an app that simplifies the booking of travel and accommodation of my employees	they can book everything in one place	Low
27	manager who oversees whereabouts of my subordinates	have an app that would grant us managers access to our employees work travel details	know exactly what my subordinates are doing.	Low
28	employee who travels a lot	have a review system	can see what others who are part of the company think of the transport and accommodation	Low
29	employee	book flight	arrive at my destination faster	Lowest
30	employee who travels overseas	book transportations and hotels in other countries	ensure I arrive there on time	Lowest