1 Requirements

Suggest some key requirements for each category (functional, and non-functional: data, environmental, user characteristics, usability goals, and user experience goals) for each of the following situations:

(a) A taxi booking app for use in a city like Cambridge

- Functional

  The app will show users a map of the nearby area and the location of all the taxis which are nearby and available to book. It will then allow users to select, order and pay for a nearby taxi through the app. The app will also allow users to see the cost of their trip before ordering it; and for the taxi driver to see how much they will be paid before starting the order. Users should be allowed to leave reviews and see previous review of their taxi drivers. Taxi drivers should also be allowed to leave reviews of their customers. The app should have some basic features preventing users with bad ratings making new accounts, for example by only allowing emails or cards to be linked to a single account. The geolocation and costs should be accurate. Users should not have to wait for hours for a taxi to arrive. The app should work for both apple and iOS .

- Data

  The primary data goal is that data collected shall conform with the GDPR . The app shall store all payment information for both the users and the drivers so that users can pay and that the drivers can receive payment. The app should store basic information about customers and drivers, for example the customer's rating and name; the driver's name, car model (and capacity), rating, driving licence and any points on their licence or criminal record. The app should store historic trip information – both the payments for accounting reasons and the trips for safety records – if someone goes missing we know who they were with at that time! We can also use this for data analysis – what cars do people like? What will they pay for etc.

- Environmental

  The app may be used in public. It should therefore not display any personal details on the homepage or any similar screen. The app may be used on 4G or poor Wi-Fi and so should not require good Wi-Fi to work properly. It may be used either by individuals or groups.

- User Characteristics

  Users may be in a rush, they may be intoxicated or may not be technologically adept. In all of these cases, our app should be easy and quick to use.

- Usability goals

  All actions should be intuitive and easily learnable. The app can be made more learnable by following established conventions, for example the settings should be in the top left, user profile should be in the top right etc. Users should not be able to accidentally order taxis: it should be very obvious what you're doing, and you should not be able to accidentally order taxis!

- User experience

  Users should find using the app easy and not at all frustrating. As part of the second part, the brightness of the interface should be consistent and users should have to click on a small number of buttons to do anything (except possibly pay). Users should have regular feedback given to them. For example: "the taxi is on it's way – it's on Kings Street and will arrive in 3 minutes"!

(b) An air traffic control system for scheduling takeoffs and landings in a large airport

- Every part of the control system must be thoroughly tested and the system must contain no known bugs. The system should allow for the easy coordination of aircraft while being fast and secure – nobody should be waiting for any meaningful amount of time on the system, and nobody should be able to schedule or delete flights without appropriate permissions. The system should also have backup servers so that it never has any downtime. The system should have facilities so that things can easily be rescheduled in an emergency.
- Data

  The system will keep all records of all flights and all logs in perpetuity. We should never delete records relating to flights – people need to check them for border checks such as for immigration or in a manhunt. The system should also record all logs of who did what for an extended period of time – if something goes wrong, it might go *very* wrong – so we need to know who did it.
- Environment

  The system will be used in a highly professional environment by people who have received specialist training. It will only be used in an office. The system will often be used in normal circumstances and will be turned on for many months at a time. The system may suddenly need to be used in a life-threatening emergency.
- Usability Requirements

  The system should be able to provide advanced users with all information quickly. It should be generally usable – and should match similar systems at other airports (we don't want to undergo extensive retraining for new staff).
- User Experience

  This system should not be stressful to use – in an emergency users will be under sufficient stress. The system should require a minimal number of clicks and inputs to present the desired information. Users should not have to *fight* the system to do their jobs.

2 User Research

For each of the user research methods below, give two concrete examples; first, of a software project where the respective method would be suitable for use and would generate meaningful and useful data (briefly describe at which stage of the iterative process you would be using the method, how the data gathering would take place and what kind of data you'll be collecting); second a software project where they would not be very suitable to use (briefly describe why this is the case).

- Questionnaires

  This is suitable initial research for most projects. Consider for example designing a system for the NHS on which to see health prescriptions, conditions, mediation etc. The demographic which would be using this app is large and diverse. We would benefit from finding out more about them – what information do they want to see quickly? What information don't they care about? Which features matter to them? This would be done in the initial research when deciding which features to incorporate.

  It's unsuitable if we're building very specialist equipment for a small group. Consider making a database system for a company. Here, we have a few strict criteria which the system must do (outlined in the specification) and otherwise we do some research to make it more usable. However our target user group is so small and personal that questionnaires would be too rigid and impersonal! We would be far better interviewing

users – there's only 10 of them! We may also not understand how the system would be used and ask the wrong questions – giving use near-useless information.

- Interviews

  Interviews are suitable for almost all projects. In fact most projects which fail, do so because the desingers do not interview users or employees. An example of an system which would benefit from this is an online retailer. For example we could use interviews in designing a university system which displays revision resources etc. We would benefit from interviewing students to find out which resources they're interested in and what features would be used most. We would interview students and staff near the end of the development process and ask them what features they care about.

  This would be unsuitable for an online retailer – customers purchase a small range of the possible products and so any manageable set of interviewees would never be reflective of the whole set of customers – in addition the retailer would be able to find out which products are selling well and put them in more prominent positions.

- Ethnography

  This would be suitable for a system used by NHS nurses and doctors. For example a controller for medical devices on their computers (rather than on individual device interfaces). We would watch nurses and doctors work on patients and see what they use, what annoys them about the current system and how it could be improved. This would take place during the initial research phase.

  Unsuitable for building an interface for a life support system on the ISS . Going to observe action on the ISS would be impractical (the cost would almost certainly be higher than the total software budget) and the success criteria are immediately obvious without observation of use.

- Lab-based observation

  Suitable for projects which you know how they will be used immediately. For example if we were developing a CAD system, we could use lab-based observation. This would happen in the middle of the development cycle after we had a working prototype.

  This is also very good for situations which don't occur naturally or occur infrequently. For example an alarm system to react to a disaster.

  This is unsuitable for research into something that's particularly sensitive. For example if we were researching peoples health habits for an app, we should not do this in the lab as people will likely not be wholly truthful.

- Focus groups

  Users can be influenced easily to fit in. This is a very, very bad strategy if you can talk to multiple people.

  However, it can make some people feel more comfortable.

  It can allow you to talk to more people if you have limited time.

  The classical example of focus groups is near-production TV shows and movies – it takes them many hours to watch the show and most of it is passive – it would make no sense to have one interviewer per interviewee.

- Card sorting

  Good for trying to figure out the layout of things in a navigation bar or in a supermarket.

  This is good for finding out which associations people make and therefore where to place things to make them as easy to find as possible.

  Card Sorting is pretty niche, it's *only* good for finding out which associations people make. We couldn't use it to find out what matters to people or any metrics such as

that! For example, it wouldn't make sense to do card sorting if we were designing a weather app – we want to know what people care about, not what groups they would place certain things in.

3 Participatory Design approaches

The course takes a User-Centred Design approach to Interaction Design. Another approach is Participatory Design (or co-design), where one or more users join the design team and are actively involved in the design (sometimes described as the product being designed *with* the users, rather than *for* the users).

(a) Would you expect that the design methods will need to change? If so, how?

All of them since we are using a different algorithm? What more can I say?

In the following, I will assume that the user has joined the design team and so we are no longer doing *anything* which is not pure Participatory Design! In reality we could use a method which was a combination of the two forms of design and get the best of both worlds!

- User research changes greatly! The user just voices their opinion on whatever we're talking about when we get onto it. We don't particularly need to ask them a ton of questions.
- We are likely to still study documentation and research similar products – this is unrelated to the user themselves and so should be unaffected by the presence of a user – if someone else has thought of something smart (which we're allowed to use) then why shouldn't we use it? Also we have to conform to industry standards and may find some key and otherwise impossible-to-think-of things if we study the documentation properly.
- Data analysis and interpretation no longer happens since we didn't really do the same sort of user research. Now we don't have a data driven approach – we have a much more personal style.
- Stakeholder mapping is unchanged; that whole part of the process hasn't changed at all – one user doesn't cover every stakeholder.
- We no longer have to do any personas since we have the user right here.
- Scenarios are sort of the same? We still need to think about how users will use the app – however now we can just ask the user who's right there how they would use the app and work like that.
- Journey mapping is changed. Now we just ask the prospective user in the design team what they'd do under these situations.
- Requirements analysis is changed. Now rather than using the data driven approach from the user research we did in the earlier part, we basically ask the user on the design team what they want and incorporate that into our requirements. We still need to establish the same requirements.
- Design process and prototyping is not changed greatly, the only difference is that there is now a user who is on the design team.
- Exploring the design space is largely unchanged.

(b) What might be some of the benefits of having users participating in the design?

Programmers can't misunderstand what the user wants.

Users get to have much more of a voice in the development process.

We get a much, much deeper understanding of what users want.

Once users start to know the designers personally, they're less likely to be "intimidated" into voicing agreeing views. For example:

interviewer: What do you think of our new product that we've spent 3 years working on.
interviewee: *Thinking it's pretty bad*; yeah it's really quite good, I like it.

We no longer have the whole expensive interview process, this can save a substantial amount of money.

Much more personal.

(c) What might be some challenges that arise from such a setup? How would you go about addressing them?

Not data driven

The users can try to control the project too much or annoy the programmers or just make the whole development process significantly slower.

"Bossy" users may annoy the design team or try to force design decisions which are impractical. For example "this nutrition app should automatically figure out the carbon footprint of the food you've just eaten and suggest healthier, lower carbon alternatives". Accurately working out the carbon footprint and then *generating* suitable alternatives may be a larger project than the rest of the nutrition app.

We have to hire a prospective user for the whole development process. This can be very expensive and is unsuitable if user time is low. For example if we were making a system for use by senior politicians, we could not have a senior politician with us throughout the design process!

We can only have meaningful input from a very limited amount of users with us during the design process. This may not be representative of the users as a whole.

Once the user starts to *know* the system too well, they no longer become a representative user and the value of their opinion decreases.

Much of the time that we develop the system, we're not making important design decisions. In these cases user input is pointless and distracting – they don't care what particular shade of white this button is.

The user may be totally inexperienced in design processes and may not understand what we're doing or may simply get in the way. This may make them feel out of their depth and they may not give proper input anymore.

The user will be unable to make any meaningful input during the prototyping stage since they're not a programmer.