YETI PRESENTS

Unlocking Connectivity:

The Ultimate Guide to IoT Software Development

UNLOCKING CONNECTIVITY

The Ultimate Guide to IoT Software Development

Robot vacuums. Smart thermostats. Self driving cars. The world is currently witnessing a profound transformation in the way we use and interact with everyday objects - a revolution driven by the Internet of Things.

The IoT (Internet of Things) is a groundbreaking concept that involves connecting everyday objects to the internet and enabling them to collect and exchange data, providing users with an unprecedented amount of functionality and automation.

While the general public often associates IoT with well-known devices like Fitbit fitness trackers and Nest smart home appliances, it's important to recognize that **this transformative technology is steadily making its way into uncharted territories** - and at the moment, we're only scratching the surface of IoT innovations.

In fact, the IoT promises a future of unprecedented possibilities and remarkable advancements across diverse domains - from smart cities to smart healthcare, to smart energy management and beyond - the IoT is reshaping our world in ways we've never imagined.

The IoT is rapidly reshaping the way we interact with everyday objects and, in turn, is providing many with remarkable opportunities for transformation and growth.

To help you harness the full potential of this extraordinary revolution, we've put together this guide with the aim of sharing a comprehensive understanding of IoT technology, and the multitude of ways it can elevate your products and business.

From initial conceptualization to final deployment, we'll cover each step of the IoT software development process, providing you with the tools, knowledge and insights that will allow you to build connected products that people love to use.

We'll also explore several strategies for optimizing user experience, provide vital insights for building successful connected software, discuss the biggest pitfalls we've encountered, and much more.

With this guide, we invite you to embark on an exciting exploration of the IoT design and development process. Get ready to unlock your products' full potential through the transformative capabilities of connected software.

THIS GUIDE COVERS

01 IoT Software Development Explained

02 Understanding UX Design

03 Connected Technology in Action

04 Vital Insights for Iot Software Development

05 Tips for building an IoT app that users love

06 Common Pitfalls in IoT Software Development

IOT SOFTWARE DEVELOPMENT EXPLAINED

Understanding the Internet of Things (IoT)

The Internet of Things, often abbreviated as IoT, is a term that has gained widespread recognition over the past decade. At its core, IoT refers to the interconnection of everyday objects or "things" to the internet. This connection allows for the collection and exchange of data - which can then be used for a multitude of purposes such as automation, enhanced user experience, monitoring, analytics and so much more.

The range of items that can be used and incorporated into the IoT is vast, ranging from everyday household appliances and wearable gadgets to complex industrial machinery and even vehicles. Essentially, any object that could be improved, or which has the potential to elevate user experiences through connectivity is a prime candidate for integration into the IoT ecosystem.

How IoT Works

When an innovator identifies an opportunity to enhance an everyday item through connectivity, their first step is devising the means to achieve it. Generally, this happens through the integration of a network of sensors, actuators, communication technologies, and other essential components within the device itself.

Here's a simplified breakdown of the IoT works:

Sensors

IoT devices are equipped with various sensors that can collect data from their surroundings. These sensors can include temperature sensors, motion detectors, GPS receivers, and more. These sensors continuously gather information, which is then processed and transmitted.

Think of a Fitbit, that has sensors to detect heart rate, skin temperature, steps taken etc.

Data Processing

The data collected by the device is then processed either locally within the device or sent to a central server for analysis. This analysis can involve identifying patterns, detecting anomalies, and making decisions based on predefined rules or machine learning algorithms.

In the example of the fitbit, the biometric data collected is used to determine many things, including calories burned, hours slept, miles walked etc.

Communication:

IoT devices use various communication protocols to transmit data to central servers, other devices, or even directly to users. Common communication protocols include Wi-Fi, Bluetooth, Zigbee, and cellular networks. The choice of protocol depends on factors such as range, power consumption, and data volume.

Centralized Control

Centralized systems, often hosted in the cloud, receive and process data from multiple IoT devices. These systems can monitor the status of devices, trigger actions, and provide users with real-time insights through user interfaces or mobile applications.

Actuators

In addition to collecting data, IoT devices often include actuators that allow them to take actions based on the processed information.

For example, a smart thermostat is able to adjust the temperature of a room based on occupancy and user preferences.

The Importance of a Well Designed IoT App

While the hardware components of IoT devices are essential, it's the software that brings the devices to life and allows them to fulfill their potential, playing a crucial role in shaping the capabilities and functionality of IoT products.

Think of IoT software as the device's brain - it makes sense of the data collected and presents it to the user in several meaningful ways.

Given that the quality of an IoT device’s software significantly shapes the users experience, it’s crucial that the highest priority is placed on creating quality software with thoughtful UX design.

IoT Apps are important because they:

1. Serve as a User Interface

Connected devices rely on software to serve as their user interfaces, whether through mobile apps, web portals, or device displays. A well-designed user interface is essential for users to interact seamlessly with the device, control its functions, and interpret data it generates.

A well-designed user interface ensures that users find a product or service easy to use and enjoyable, and enhances the usability of the connected device. This means that users can interact with it efficiently, accomplish tasks easily, and navigate without confusion - reducing frustration and increasing user satisfaction.

2. Provide Effective Data Collection + Presentation

Most connected devices collect and provide data to users, so when creating its companion software it’s important that it provinces features that effectively utilize this data. For example, a smart fitness device might collect activity data, which the software can then use to analyze progress, set goals, provide nutritional guidance and much more.

However, it's not enough for the software to be proficient at data analysis alone.

Equally critical is the user interface's capability to **present this data in a manner that is lucid, comprehensible, and conducive to action**. This can be accomplished through a variety of visualization methods, including graphs, charts, data points, and more, empowering users to extract valuable insights and make informed decisions rooted in the information at their disposal.

3. Ensure Accessibility

When creating your connected software, you'll want to ensure that its interface is accessible to all users, including those with disabilities. Not only is this a matter of inclusivity, it's also a matter of compliance with accessibility standards. Many companies have been sued over non-compliance with ADA standards - 10,982 suits filed in federal court in 2020 to be exact (learn more about ADA compliance in our free guide).

Good IoT software development and design considers accessibility from the outset, making the device usable by a broader audience. This includes ensuring correct color contrast, ensuring alt text is included with all images, fully accessible multimedia and much more.

4. Are Secure

Good IoT software development practices are essential for securing connected devices as vulnerabilities in the software can expose devices to cyber threats. The following are a few key areas when building secure and privacy-focused IoT software:

- **Data encryption and secure communication protocol**s: To keep users data safe, it is crucial that you implement robust encryption mechanisms, such as industry-standard algorithms that encode sensitive data and render it unreadable to unauthorized parties.

- **Secure user authentication and access controls**: To prevent unauthorized access to connected devices and the data they store, implementing secure user authentication mechanisms such as multi-factor authentication, should be employed to verify the identity of users. Access controls additionally bolster security by assigning appropriate privileges based on user roles, ensuring that sensitive functionalities and data are only accessible to authorized individuals.

- **Protection against vulnerabilities and potential cyber threats:** It's important to ensure that your IoT software development team utilizes regular security assessments and code reviews to help identify and address any security loopholes or weaknesses. Following secure coding practices, adhering to industry security standards, and staying informed about emerging threats are key measures to safeguard against cyber attacks, data breaches, and unauthorized access.

5. Allow for Cross-Device Integration

Users no longer interact with just one device - and they expect a seamless experience across all of their various gadgets and platforms. Good IoT software development acknowledges this trend and prioritizes compatibility and integration. This means that a user can start a task on one device and continue it on another without any disruptions. For instance, you might begin reading an article on your smartphone during your morning commute and then seamlessly switch to your tablet or computer when you reach the office.

Furthermore, cross-device integration plays a pivotal role in the Internet of Things (IoT) ecosystem, where devices from different manufacturers and categories need to communicate and work together effortlessly.

For example, your smart home thermostat should communicate with your smartphone, allowing you to adjust the temperature remotely. The user should also be able to integrate it with voice-activated assistants like Amazon Alexa or Google Assistant. In this context, good software development is not only about ensuring that your software works on multiple devices but also that it interacts seamlessly with a diverse range of devices, services, and platforms, ultimately enhancing the user's experience and convenience.

6. Perform Well and are Efficient

Performance and efficiency are paramount in IoT software development. Efficient software ensures that connected devices operate seamlessly and respond promptly to user interactions. This aspect is especially critical for real-time applications and IoT devices, where any latency or delay can significantly impact the user experience and the device's overall functionality.

In the realm of IoT, where devices often interact with one another in intricate ways, performance and efficiency are essential for timely and reliable data processing. For example, in a smart home, if a motion sensor detects movement, it must communicate this information to other devices (e.g., lights or security cameras) without noticeable delays.

Efficient software ensures that these interactions occur swiftly and that data is processed and acted upon in a timely manner, contributing to a seamless and responsive user experience.

Energy efficiency is another critical aspect of IoT software development, especially for battery-powered devices. Optimizing code to minimize energy consumption prolongs the device's battery life, reducing the need for frequent recharging or battery replacements.

UNDERSTANDING UX DESIGN

Every day over 5000 apps are submitted to the app store. Some are successful, while many fail to thrive. What are the differentiating factors in these cases?

The truth is that creating a successful IoT app isn't just about building something that "works". What users really want is an app that works well and feels good to use. Having a positive experience with an app is the result of great UX (user experience) design.

UX design is all about optimizing your users' experience by understanding their needs, and creating an app that will allow them to meet those needs in the easiest and most pleasurable way possible. Great UX design is born from extensive research - so it's important that you ensure your development team is covering each of the following steps.

Getting to know your user

Getting to know your user is one of the most crucial steps in the IoT design process, as understanding your users needs, wants, and the problem your app will be solving for them will allow you to create an app experience that meets those needs in the easiest and most pleasurable way possible.

Unfortunately, we've found that some development firms cut corners when it comes to this stage of the process, and sadly, the results are nearly always the same - an app that functions, but is miserable to use.

To design an app that users love, be sure to complete each and every one of the following steps.

1. User Interviews

User interviews will allow your team to uncover the underlying motivations and potential biases of your app's users. These interviews will also lay the groundwork for the user personas you'll be creating later on in your design and development process, so be sure not to skip them.

To begin this process, **get together as a team to brainstorm exactly who your product's ideal user will be.** Consider things like their job,age, the apps they currently use and the problems they have that your app will solve. Doing a bit of brainstorming around these topics should **give you a pretty good idea of the kind of people you'll want to interview.**

Once you have a good understanding of who your user will be, it's time to create a list of relevant questions designed to provide you with deep insight into the user and their wants and needs. Try to create questions that will help provide:

- Feedback on your app idea
- Information on the demand for your app
- Suggestions on how to improve your idea
- Insight into features real users want

Once you've developed your list of questions, you're ready to find interviewees. Refrain from interviewing co-workers, friends and family, as they will generally be positively biased. Instead, find individuals that you believe would use your app once it's built and interview them. There are many methods for doing so - scouting out relevant message boards and social media pages - or even real world locations where they might hang out.

2. Empathy Mapping

An empathy map is a tool that you'll use to articulate everything you know about a particular type of user- including their thoughts, emotions, and needs. You've likely gathered much of this through your user interviews.

With empathy mapping, your goal is to create a document that externalizes all of your team's knowledge of the user, which will provide you with a shared understanding of their needs.

Empathy mapping is essential for several reasons. First and foremost, it enables developers to identify and prioritize user needs, ultimately leading to a more efficient and effective design process. By considering users' pain points, desires, and expectations, developers can make informed decisions that will positively impact the user experience.

Additionally, empathy mapping encourages cross-functional collaboration within the development team, ensuring that everyone understands the user's perspective and is aligned with the project's objectives.

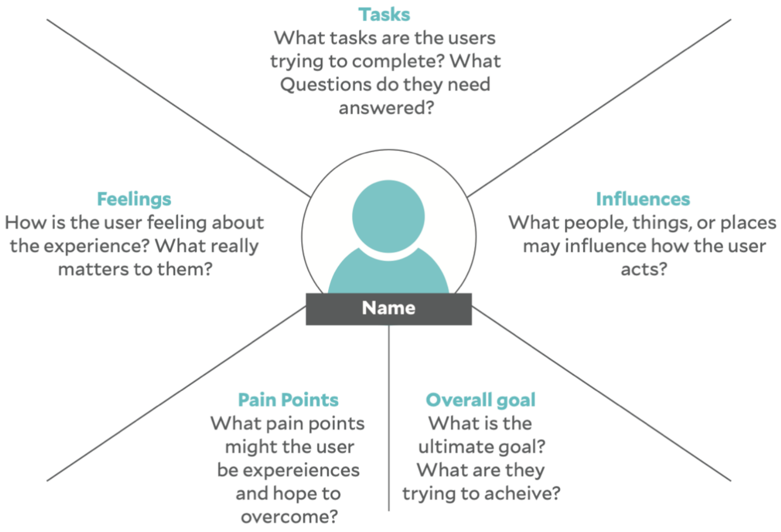
There are lot of other benefits to empathy mapping, including:

- ***Removing bias from your designs***
- ***Discovering weaknesses in your research***
- ***Uncovering user needs that the user themselves may not even be aware of***
- ***Understanding what drives users' behaviors***
- ***Guidance towards meaningful innovation***

Empathy mapping involves creating a visual representation of a user's perspective, which allows the team to step into the user's shoes and gain valuable insights into their behaviors and motivations. By prioritizing empathy, app developers can create solutions that align more closely with user expectations, resulting in a more satisfying and user-friendly product.

To conduct empathy mapping, you typically start by defining your target user persona (below), and then gather insights through user research methods such as interviews, surveys, or observations.

The next step is to create a visual representation of the user's experience, which can be done through a simple chart or diagram. The chart typically includes four quadrants: "Say," representing what the user says; "Think," reflecting their thoughts; "Do," indicating their actions, and "Feel," expressing their emotions.


Tasks
What tasks are the users trying to complete? What Questions do they need answered?
Feelings
How is the user feeling about the experience? What really matters to them?
Influences
What people, things, or places may influence how the user acts?
Name
Pain Points
What pain points might the user be expereiences and hope to overcome?
Overall goal
What is the ultimate goal? What are they trying to acheive?

Populate these quadrants with insights gathered from user research, which should help the development team better understand the user's mindset and needs.

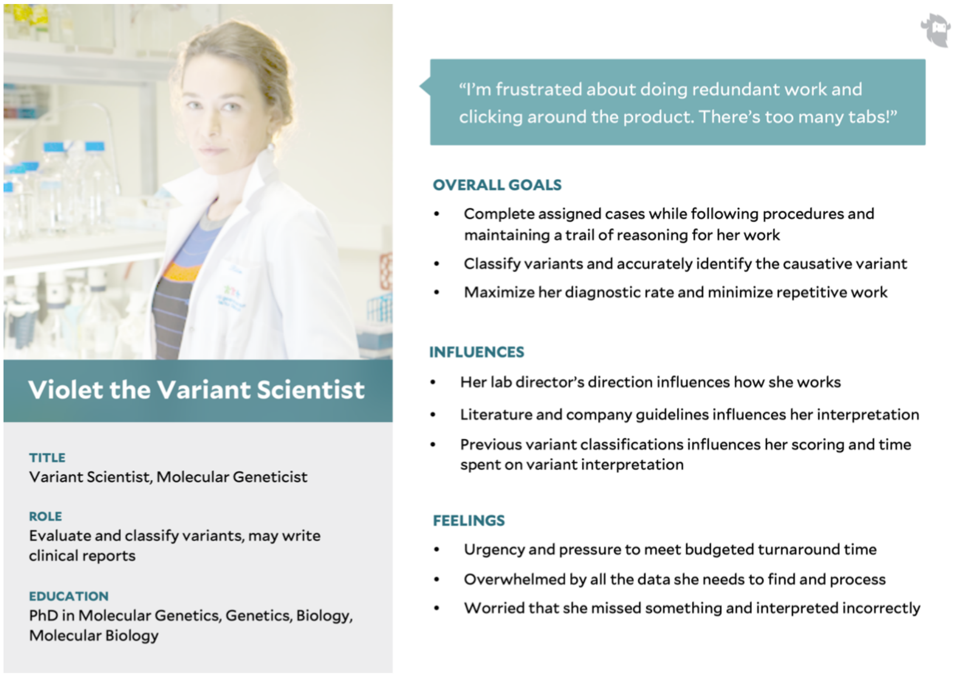
Empathy mapping should be an iterative process, and the insights gained from it should guide the design and development of the app, ensuring that it resonates with the user's perspective and delivers a superior user experience.

3. User Personas

Once your team has synthesized all of the information they've gathered through empathy mapping, they'll begin to create a User Persona. A user persona is a fictional representation of your product's ideal user. It captures what they are hoping to achieve by using your product and the various factors that may contribute to their use of the product.

Good user personas are extremely helpful in creating user-centric products. They should provide your team with insight into the users motivations and needs, and create a unified vision of exactly who you are building your product for.

They'll also inform every aspect of the app your team creates, from the language, images and designs you use to the messages you convey, so it's a good idea to dedicate some time to them.



Violet the Variant Scientist

TITLE
Variant Scientist, Molecular Geneticist

ROLE
Evaluate and classify variants, may write clinical reports

EDUCATION
PhD in Molecular Genetics, Genetics, Biology, Molecular Biology

"I'm frustrated about doing redundant work and clicking around the product. There's too many tabs!"

OVERALL GOALS

- Complete assigned cases while following procedures and maintaining a trail of reasoning for her work
- Classify variants and accurately identify the causative variant
- Maximize her diagnostic rate and minimize repetitive work

INFLUENCES

- Her lab director's direction influences how she works
- Literature and company guidelines influences her interpretation
- Previous variant classifications influences her scoring and time spent on variant interpretation

FEELINGS

- Urgency and pressure to meet budgeted turnaround time
- Overwhelmed by all the data she needs to find and process
- Worried that she missed something and interpreted incorrectly

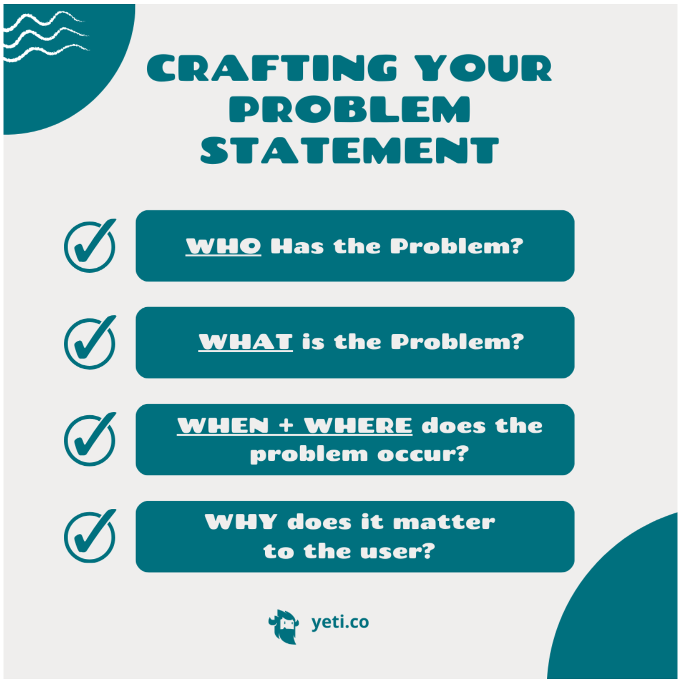
To create your user personas, begin identifying common patterns and characteristics among your potential users. This might include demographic information, behaviors, goals, pain points, and motivations. Your goal will be to take all of this information and distill it into a few fictional but accurate user personas.

Each persona should have a name, a photo, and a detailed description, summarizing their background, preferences, and goals. Creating user personas helps to ensure that the development team has a clear and humanized understanding of the people who will be using the app, ultimately guiding the design and development process to create a more user-centric and effective product.

Defining Your Problem Statement

A problem statement is a precise articulation of the user-centered issue or need that your app aims to solve or address for the user. It should outline the specific problem, its significance, and the expected impact of solving it.

Creating a meaningful and actionable problem statement is an important part of the app design process as it helps you fully understand your goal and creates a clear-cut objective to work towards. It will also help you kick start your ideation process.



CRAFTING YOUR PROBLEM STATEMENT

✓ WHO Has the Problem?

✓ WHAT is the Problem?

✓ WHEN + WHERE does the problem occur?

✓ WHY does it matter to the user?

yeti.co

UX, (user experience design), is all about solving a user problem, so establishing a clear idea of exactly what problem your app will be solving for your ideal user is a crucial first step.When considering the problem your app will be solving It can be helpful to think of the problem as an unmet need. Your app development process should be focused on designing a solution that meets this need, and your problem statement should identify the gap between the user's current state and their desired goal.

For example, “I am a young working professional trying to eat better, but I’m struggling because I work long hours and don’t always have time to go grocery shopping and prepare my meals. This makes me feel frustrated and bad about myself.”

A good problem statement focuses and gives good context around the user but is broad enough to allow room for creative design solutions. It should be specific enough though, that you’ll be able to narrow in on a suitable solution. Overly broad problem statements lead to unhelpful products.

Make sure to cover the “5 W’s” when creating your problem statement:

- **Who is experiencing the problem?**
- **What is the problem?**
- **Where does the problem present itself?**
- **Why does it matter?**

User Journey Mapping

User journey maps provide a visual representation of a user's interactions and experiences with an app throughout their entire journey, from the initial touchpoints to the final goals or outcomes.

Their purpose is to provide designers with a better understanding of the user’s perspective, providing them with insight into their thoughts and feelings throughout the process. Ultimately user journey maps are important because they help identify critical touchpoints, moments of frustration, and areas for improvement that allows your team to prioritize enhancements effectively.

User Journey Stages

Your user journey includes each of the following stages:

App discovery

How the user finds your app and why they chose it.

App download

Why, how and where did the user download your app?

App onboarding

How are users presented with the app's value proposition? How do they learn key features and provide necessary permissions?. Are you providing them with guided assistance that allows them to explore and use all of your app's features effectively?

In-app engagement

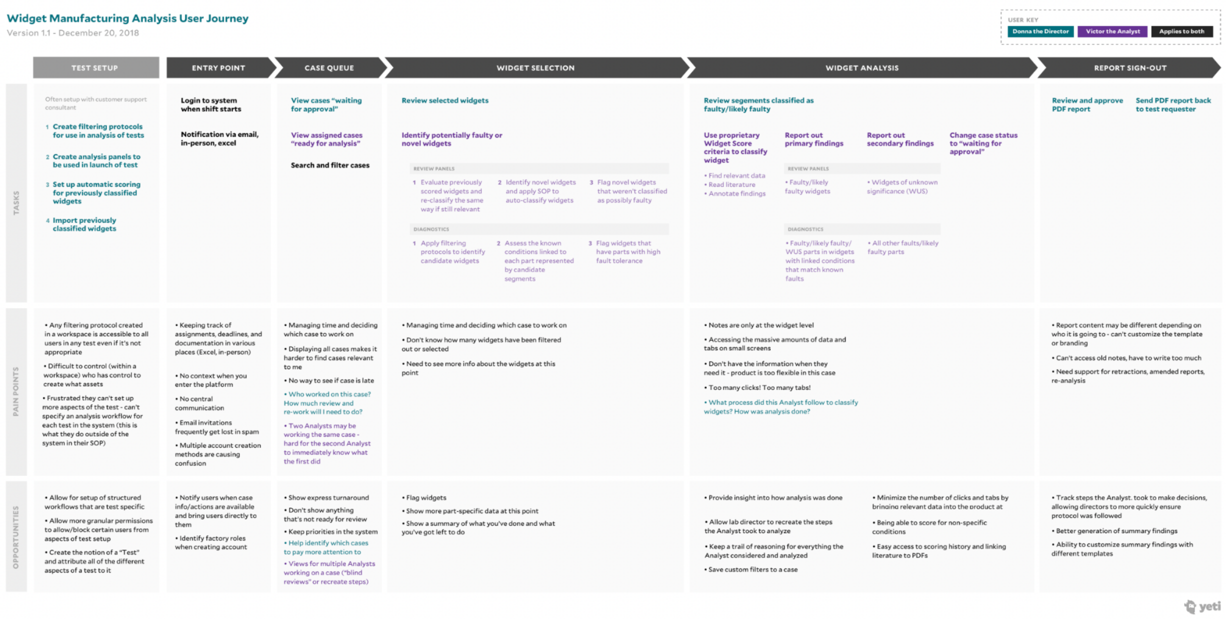
How does your app keep users engaged and loyal? Many users lose interest after the initial onboarding stage, so boosting engagement is vital for long-term growth and profitability. Do you have a strategy for nudging users to adopt key features or for increasing the number of active users on a daily, weekly, and monthly basis?

In-app purchase

For some apps, the primary objective is to encourage in-app purchases - how will your app do so? Once an in-app purchase has been made, how will you encourage repeat purchases. Will you utilize email, push notifications, and in-app notifications to provide recommendations?

User loyalty

As users progress through your app's user journey, their needs, goals, and expectations evolve. How will you ensure that users utilize your app on a regular basis? How will you encourage them to try new app experiences? How will you keep them engaged with your product?

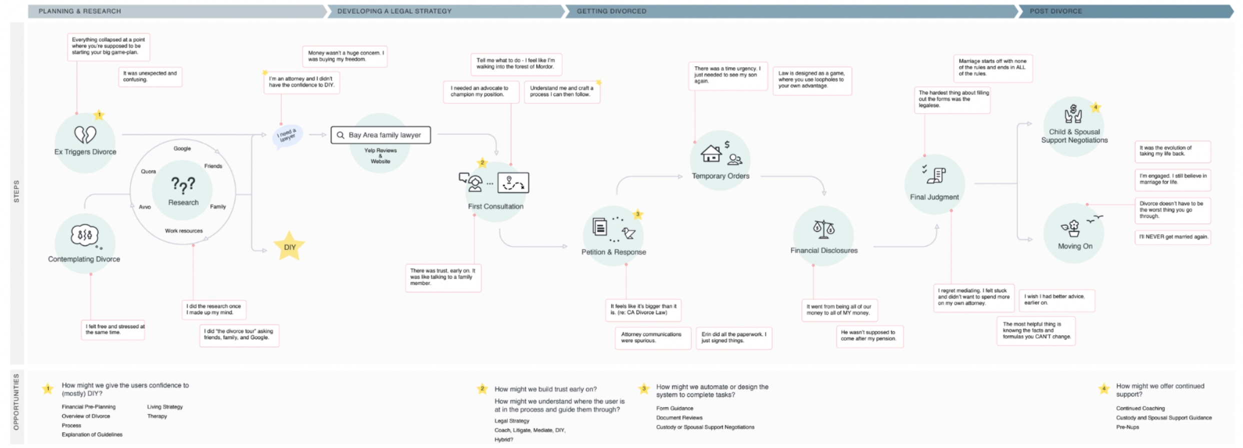


Widget Manufacturing Analysis User Journey

Version 1.1 - December 20, 2018

USER KEY: Donna the Director | Victor the Analyst | Applies to both

	TEST SETUP	ENTRY POINT	CASE QUEUE	WIDGET SELECTION	WIDGET ANALYSIS	REPORT SIGN-OUT
TASKS	Often setup with customer support consultant 1 Create filtering protocols for use in analysis of tests 2 Create analysis panels to be used in launch of test 3 Set up automatic scoring for previously classified widgets 4 Import previously classified widgets	Login to system when shift starts Notification via email, in-person, excel	View cases "waiting for approval" View assigned cases "ready for analysis" Search and filter cases	Review selected widgets Identify potentially faulty or novel widgets REVIEW PANELS 1 Evaluate previously scored widgets and re-classify the same way if still relevant 2 Identify novel widgets and apply SOP to auto-classify widgets 3 Flag novel widgets that weren't classified as possibly faulty DIAGNOSTICS 1 Apply filtering protocols to identify candidate widgets 2 Assess the known conditions linked to each part represented by candidate segments 3 Flag widgets that have parts with high fault tolerance	Review segements classified as faulty/likely faulty Use proprietary Widget Score criteria to classify widget • Find relevant data • Read literature • Annotate findings Report out primary findings REVIEW PANELS • Faulty/likely faulty widgets DIAGNOSTICS • Faulty/likely faulty/ WUS parts in widgets with linked conditions that match known faults Report out secondary findings REVIEW PANELS • Widgets of unknown significance (WUS) DIAGNOSTICS • All other faults/likely faulty parts Change case status to "waiting for approval"	Review and approve PDF report Send PDF report back to test requester
PAIN POINTS	• Any filtering protocol created in a workspace is accessible to all users in any test even if it's not appropriate • Difficult to control (within a workspace) who has control to create what assets • Frustrated they can't set up more aspects of the test - can't specify an analysis workflow for each test in the system (this is what they do outside of the system in their SOP)	• Keeping track of assignments, deadlines, and documentation in various places (Excel, in-person) • No context when you enter the platform • No central communication • Email invitations frequently get lost in spam • Multiple account creation methods are causing confusion	• Managing time and deciding which case to work on • Displaying all cases makes it harder to find cases relevant to me • No way to see if case is late • Who worked on this case? How much review and re-work will I need to do? • Two Analysts may be working the same case - hard for the second Analyst to immediately know what the first did	• Managing time and deciding which case to work on • Don't know how many widgets have been filtered out or selected • Need to see more info about the widgets at this point	• Notes are only at the widget level • Accessing the massive amounts of data and tabs on small screens • Don't have the information when they need it - product is too flexible in this case • Too many clicks! Too many tabs! • What process did this Analyst follow to classify widgets? How was analysis done?	• Report content may be different depending on who it is going to - can't customize the template or branding • Can't access old notes, have to write too much • Need support for retractions, amended reports, re-analysis
OPPORTUNITIES	• Allow for setup of structured workflows that are test specific • Allow more granular permissions to allow/block certain users from aspects of test setup • Create the notion of a "Test" and attribute all of the different aspects of a test to it	• Notify users when case info/actions are available and bring users directly to them • Identify factory roles when creating account	• Show express turnaround • Don't show anything that's not ready for review • Keep priorities in the system • Help identify which cases to pay more attention to • Views for multiple Analysts working on a case ("blind reviews" or recreate steps)	• Flag widgets • Show more part-specific data at this point • Show a summary of what you've done and what you've got left to do	• Provide insight into how analysis was done • Allow lab director to recreate the steps the Analyst took to analyze • Keep a trail of reasoning for everything the Analyst considered and analyzed • Save custom filters to a case • Minimize the number of clicks and tabs by bringing relevant data into the product at • Being able to score for non-specific conditions • Easy access to scoring history and linking literature to PDFs	• Track steps the Analyst. took to make decisions, allowing directors to more quickly ensure protocol was followed • Better generation of summary findings • Ability to customize summary findings with different templates



Everything collapsed at a point where you're supposed to be starting your big game-plan.

It was unexpected and confusing.

I felt free and stressed at the same time.

I did the research once I made up my mind.

I did "the divorce tour" asking friends, family, and Google.

Money wasn't a huge concern. I was buying my freedom.

I'm an attorney and I didn't have the confidence to DIY.

Tell me what to do - I feel like I'm walking into the forest of Mordor.

I needed an advocate to champion my position.

Understand me and craft a process I can then follow.

There was trust, early on. It was like talking to a family member.

There was a time urgency. I just needed to see my son again.

Law is designed as a game, where you use loopholes to your own advantage.

It feels like it's bigger than it is. (re: CA Divorce Law)

Attorney communications were spurious.

Erin did all the paperwork. I just signed things.

It went from being all of our money to all of MY money.

He wasn't supposed to come after my pension.

Marriage starts off with none of the rules and ends in ALL of the rules.

The hardest thing about filling out the forms was the legalese.

I regret mediating. I felt stuck and didn't want to spend more on my own attorney.

I wish I had better advice, earlier on.

The most helpful thing is knowng the facts and formulas you CAN'T change.

It was the evolution of taking my life back.

I'm engaged. I still believe in marriage for life.

Divorce doesn't have to be the worst thing you go through.

I'll NEVER get married again.

1 How might we give the users confidence to (mostly) DIY?

Financial Pre-Planning
Overview of Divorce
Process
Explanation of Guidelines
Living Strategy
Therapy

2 How might we build trust early on?

How might we understand where the user is at in the process and guide them through?

Legal Strategy
Coach, Litigate, Mediate, DIY,
Hybrid?

3 How might we automate or design the system to complete tasks?

Form Guidance
Document Reviews
Custody or Spousal Support Negotiations

4 How might we offer continued support?

Continued Coaching
Custody and Spousal Support Guidance
Pre-Nups

How to Map a Users Journey

Step 1: Establish Your Goals

Before beginning to map your user experience, it's essential to lay out your objectives. Determine whether your focus is on the entire user journey or a specific aspect of your app's functionality. Collaborate with various teams such as product development and marketing to define these objectives, capitalizing on their expertise in different facets of the app.

Step 2: Craft User Personas

If you haven't already, you'll want to create user personas - fictional representations of your target audience based on thorough research. Take into account the unique characteristics and attributes each type of user has. **(See the previous section for a more in depth explanation)**

Step 3: Identify Interaction Points and Communication Channels

Map out all the points of interaction where users engage with your app. The more touch points you include in your journey map, the more comprehensive your perspective becomes. Some touchpoints to consider include push notifications, websites, search engines, other mobile apps, email marketing, and social media.

Step 4: Visualize the User's Path

Visualize how users interact with your app, from their initial encounter to conversion. Capture their thoughts, emotions, and actions throughout this journey to understand how your app fits into their daily life and helps them solve problems.

Keep in mind that each user persona requires a distinct user journey tailored to their unique needs and challenges; hence, avoid relying on a generic template.

Step 5: Identify and Overcome Hurdles

Now that you have a grasp of the user's journey within your app, pinpoint the challenges and critical points that could impede their progression through the various stages. For instance, a user attempting an in-app purchase may encounter frustration if suddenly asked to sign up, which could lead to abandonment. Cumbersome registration procedures and unclear calls to action also present potential obstacles.

Compile a list of all these obstacles and brainstorm potential solutions. If the user journey evolves after implementing these solutions, consider creating a new journey to assess the results.

Step 6: Design and Test the Final User Journey

Consolidate all the gathered data into a comprehensive map, encompassing the users steps, success criteria for each step, retention rates, conversion rates, interaction points, obstacles, and strategies for overcoming them. However, don't stop at the mapping stage—testing the user journey firsthand is essential for a thorough understanding of the user experience and to facilitate necessary improvements based on your findings.

Step 7: Develop Key Performance Indicators (KPIs) and Measure Success

To progress toward your objectives, it's crucial to evaluate the performance of your user journey. Create distinct key performance indicators (KPIs) to assess each user journey's effectiveness. Monitor mobile app metrics such as active users, cost per acquisition, conversion rate, retention rate, and lifetime value. By analyzing these metrics, you can evaluate the efficacy of your marketing campaigns and user personas, identifying specific areas that require enhancements in your user journey.

Step 8: Continuously Enhance Your Mobile App with Fresh Data

Optimizing your app is an ongoing process. Always seek ways to enhance the user experience and cater to the needs of every user persona. This may involve streamlining unnecessary steps in the user journey or finding methods to engage users for longer periods.

Consider the use of A/B testing to fine-tune and improve your app's performance. Remember that optimization is an ongoing journey, and you should continually collect insights, analyze data, and refine your app to create an even more exceptional user experience.

Ideation

Now that you've gathered all of the pertinent information about your ideal user and the problem your product will be solving, you're ready to begin narrowing your strategic focus and generating solutions with your team.

This phase of your development process is all about generating ideas - you've chosen a goal and are now ready to approach it from every angle possible. During this stage, your objective is to uncover hidden insights and generate as many potential solutions as possible.

During this phase you should set up an ideation workshop where you and your team will brainstorm as many ideas as possible, ultimately sketching out what the finished product will do and how it will do it.
To start, you'll want to begin generating solutions for one of the pain points you've identified in your user journey map. There are a few different exercises that will help in this process - we find "Crazy 8's" to be particularly helpful:

Crazy 8's:

The intention of this exercise is to generate as many ideas as possible within a short time frame - the emphasis here should be on quantity over quality of ideas. By giving individuals 8 minutes to generate 8 ideas you help push them to move past their first idea, which is frequently the least innovative.

Begin this exercise by handing out sheets of paper to each individual participating. As the team to fold the paper in such a way that it creates eight sections.

Set a timer for eight minutes - and ask team members to sketch one idea in each of the eight sections. Remind everyone that these ideas are just sketches - they don't have to be works of art, but should represent the app solution they think will solve the users problem.

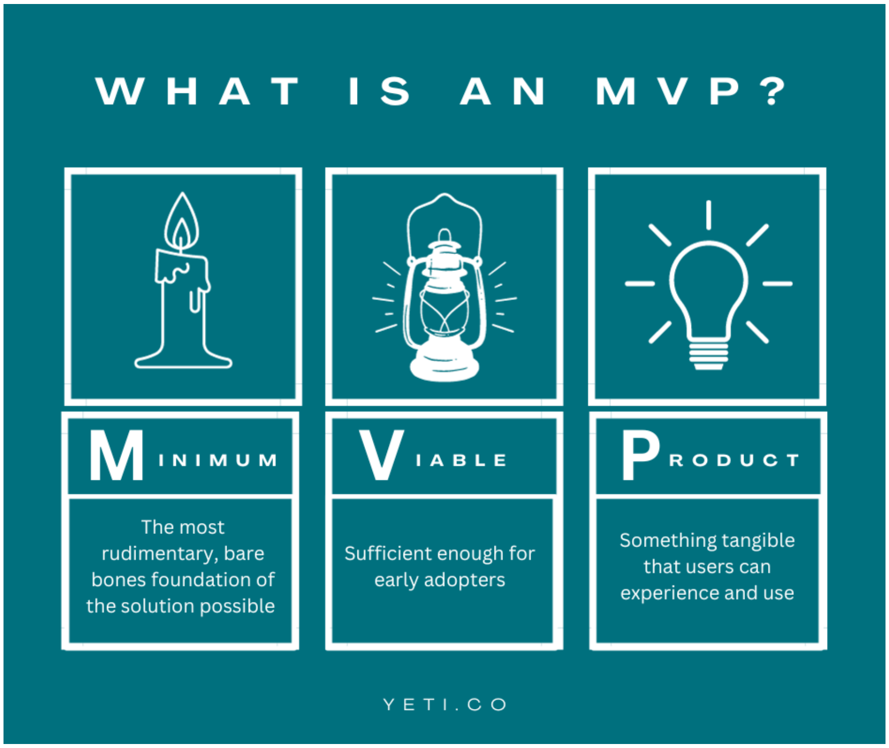
To help team members who don’t have a background in design, you might want to hold a quick how to sketch session. It's also important to remind everyone that their ideas don’t have to be great - this exercise should be about quieting everyone's inner critic and allowing the creative juices to flow.

Once your team has completed their individual ideation process, allow each individual to present their ideas to the team. Allow them to explain how their proposed solutions work and any other valid information.

Once everyone on the team has presented, tape all the sheets to a wall and then ask the team to” dot vote” for their favorites. This can lead to more discussion or even more sketch brainstorming.

Your MVP

MVP stands for Minimum Viable Product, and it’s essentially a very scaled down version of your product. **Your MVP will consist of only the most essential features necessary to attract early- adopters or funding and validate your product idea.**



WHAT IS AN MVP?

Minimum: The most rudimentary, bare bones foundation of the solution possible

Viable: Sufficient enough for early adopters

Product: Something tangible that users can experience and use

YETI.CO

Many people are surprised by the fact that we don’t recommend attempting to build an entire app in one fell swoop - not only would this be extremely time consuming - it can also be extremely risky.

Starting out by building an MVP allows you to release your product early in the development process, which helps to determine early on whether it actually has the potential to succeed.

For example if, upon releasing your MVP, you find that users are excited about your app you can start working on a full production version with some assurance that your final product will be successful.

On the other hand, if your MVP fails to make an impact on users, you might decide to pivot and take your product in a different direction - or you may realize that it simply isn’t a good idea to see it through to a full production version. In this scenario, the time, money, and effort saved is usually huge.

Defining your MVP is all about determining exactly which features the first iteration of your app will include - so you'll want to join your team for a workshopping session in which you look back at all of the work you’ve done leading up to this point, and compile a list of important app features to determine exactly which features need to be included in this first iteration.

The steps for determining what you MVP will include:

1. Revisit Your Goal

Start by clarifying the primary goal of your app, including the problem it solves, and the value it will provide to users? Understanding your app's core purpose is essential to creating your MVP

2. Revisit User Needs:

Look at all the user research you've already completed and revisit their needs, pain points, and preferences. This will help you prioritize features that address the most crucial user requirements.

3. Prioritize Features:

Prioritize your apps features based on their importance to achieving the app's primary goal, and how well they meet user needs. While some features are core to your app's functionality (for example, a shopping cart for an e-commerce app), others are just nice-to-have (ex. pop-ups directing users to specific features)

4. Create a Feature List:

Develop a comprehensive list of potential features. Categorize them into essential features, additional features, and features that can be deferred to later versions.

5. Consider Technical Feasibility:

Assess the technical feasibility of each feature. Some features may require significant development effort and could slow down your MVP release. Focus on features that can be implemented efficiently.

6. Build a Lean Prototype:

Create a simple, functional prototype of your app with the essential features. This prototype should demonstrate the core functionality of your app.

7. Test and Iterate:

Test your MVP prototype with a small group of users to gather feedback. Use this feedback to refine and enhance your app. This iterative process ensures you're building what users truly need.

8. Set Release Criteria:

Define criteria for when your MVP is ready for release. These criteria might include the stability of core features, user feedback, and performance benchmarks.

9. Launch and Gather Data:

Once your MVP is released, collect data on how users are interacting with the app. This data will inform your decisions for future feature development.

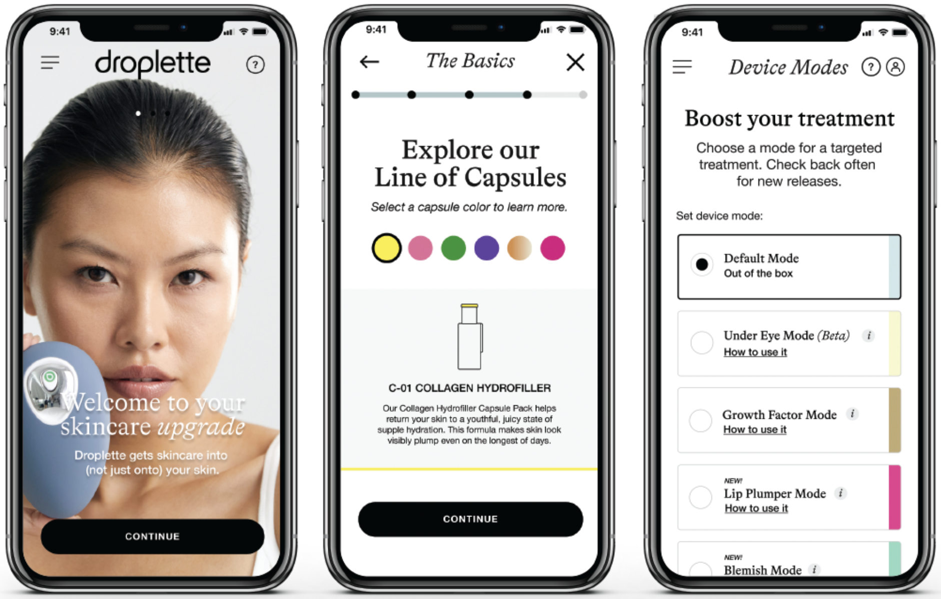
10. Plan for Iterations:

Remember that the MVP is just the beginning. Plan for iterative releases, where you gradually add additional features based on user feedback and evolving requirements.

CONNECTED TECHNOLOGY IN ACTION

Droplette

Droplette is a simple to use connected skin care device that delivers serums through micro-infusion technology. This device pairs with the Droplette App to provide users with an unparalleled at-home skin care routine.



Welcome to your skincare *upgrade*

Droplette gets skincare into (not just onto) your skin.

Explore our Line of Capsules

Select a capsule color to learn more.

C-01 COLLAGEN HYDROFILLER

Our Collagen Hydrofiller Capsule Pack helps return your skin to a youthful, juicy state of supple hydration. This formula makes skin look visibly plump even on the longest of days.

Boost your treatment

Choose a mode for a targeted treatment. Check back often for new releases.

Set device mode:

● Default Mode
Out of the box

○ Under Eye Mode *(Beta)*
How to use it

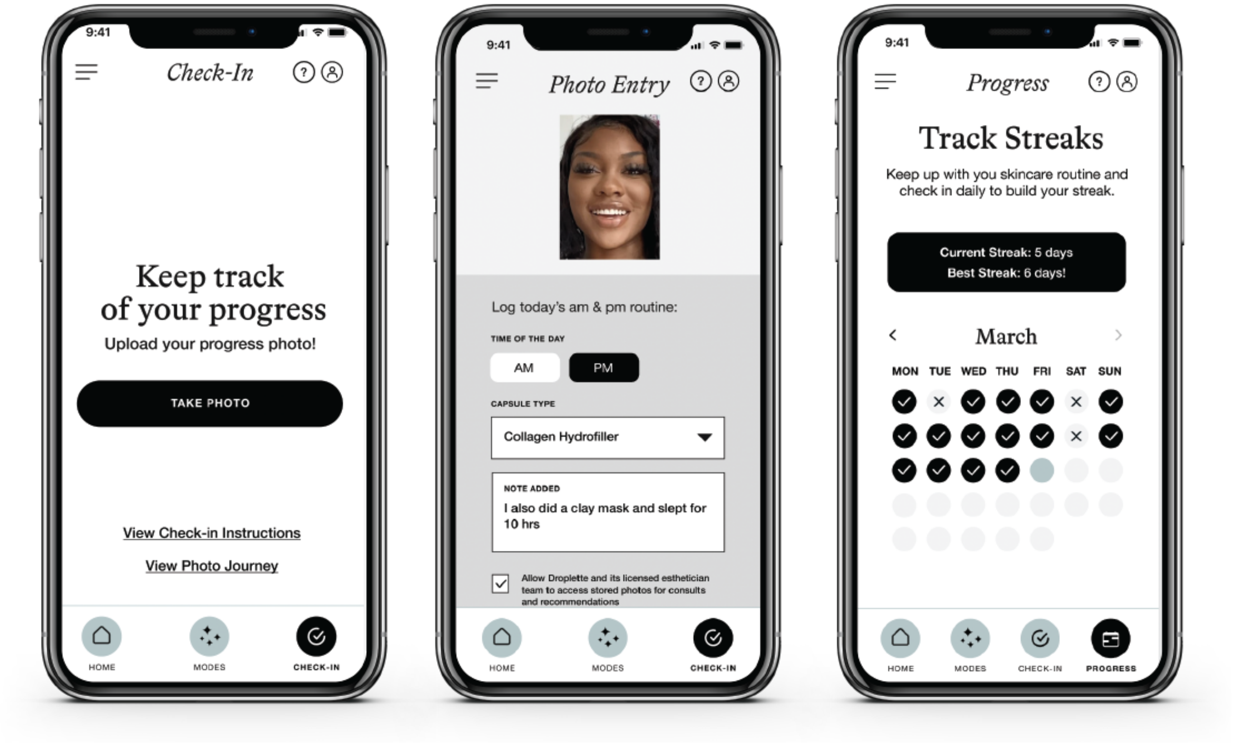
○ Growth Factor Mode
How to use it

NEW!
○ Lip Plumper Mode
How to use it

NEW!
○ Blemish Mode

On its own, the Droplette device provides users with incredibly effective skin care - but paired with the Droplette App, the user's experience is elevated to an entirely new level. While acting as the device’s control panel is its most obvious function, the Droplette app’s magic is actually in the way it serves as a source of motivation and inspires users to continue utilizing Droplette’s serums.

The Droplette app’ regimen builder provides users with several visual reminders of their progress, most notably, a daily photo journal. By providing users with the ability to take and compare daily photos, the app displays a clear visual of the user's progress - an incredible source of motivation that inspires users to continue with their Droplette routine.



Check-In

Keep track of your progress

Upload your progress photo!

Photo Entry

Log today’s am & pm routine:

TIME OF THE DAY: AM [✓ PM]

CAPSULE TYPE: Collagen Hydrofiller

NOTE ADDED: I also did a clay mask and slept for 10 hrs

☑ Allow Droplette and its licensed esthetician team to access stored photos for consults and recommendations

Progress

Track Streaks

Keep up with you skincare routine and check in daily to build your streak.

Current Streak: 5 days
Best Streak: 6 days!

March

MON	TUE	WED	THU	FRI	SAT	SUN
✓	✗	✓	✓	✓	✗	✓
✓	✓	✓	✓	✓	✗	✓
✓	✓	✓	✓			

Users are additionally provided with a journal feature, allowing them to track exactly which serums they are using and the results they provided. To create a seamless purchasing experience, users are also able to purchase new serums directly from the app.

To help keep them on track, Droplette app users also receive personalized 1:1 feedback and advice from certified aestheticians directly through the app, providing users with helpful advice, and driving serum sales

9 VITAL INSIGHTS FOR IOT SOFTWARE DEVELOPMENT

Connected devices have become ubiquitous in today's world, ranging from fitness wearables to smart home appliances, this technology is revolutionizing the way we interact with everyday objects.

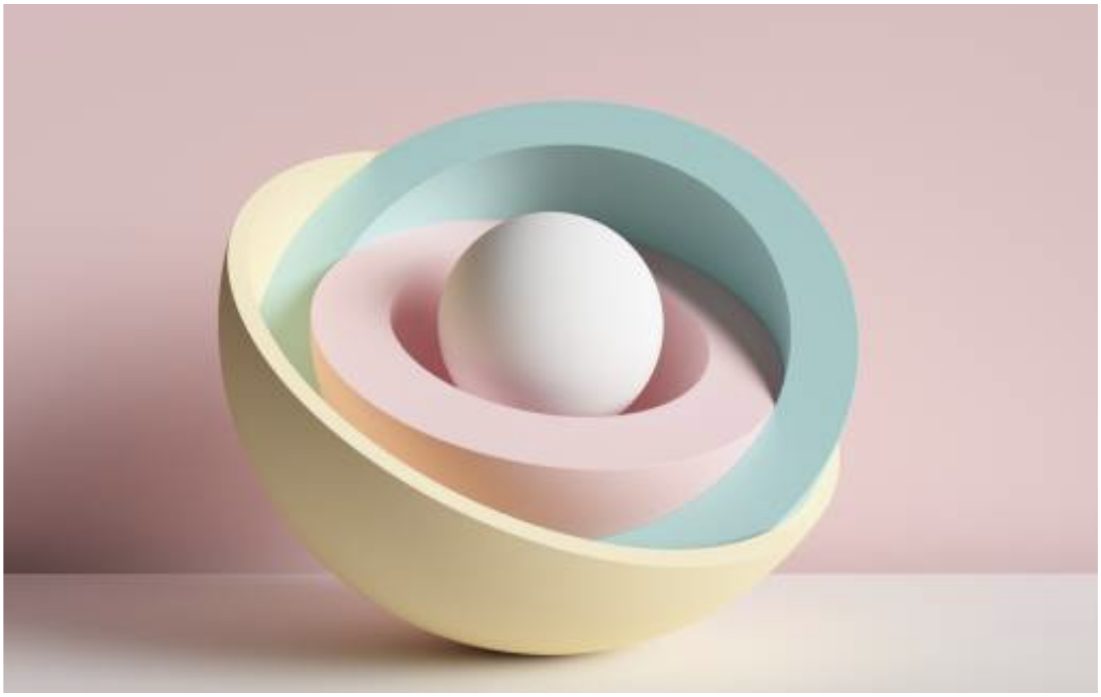
However, amid the spotlight on these devices, there is an often overlooked but vital component of the connected ecosystem - the connected app and IoT software development.

While physical devices rightfully garner attention, it’s important to recognize that the software that drives it is equally significant. Whether it's a smart home gadget, wearable technology, or industrial IoT machinery, connected apps and software serve as the brains behind the device's functionality and interactions. They are the keys that unlock a connected device's full potential, enabling a seamless and user-friendly experience.

When creating a connected product, a common tendency is to concentrate primarily on crafting an innovative, user-friendly, and feature-rich device - with the app that powers it receiving only secondary consideration. In our experience, this approach is a significant misstep as in the majority of cases, much of the user's direct experience with a connected product is with the app itself.

When building a connected app, it's crucial that you strive beyond mere functionality. Instead, your focus should be on creating a streamlined, user friendly, feature rich and aesthetically pleasing app that people love to use.

Below, we outline the most crucial elements to consider during the IoT software development process. Ensuring that each of these areas are carefully considered throughout your app build will allow you to build a connected product that users can't get enough of.



1. Compatibility and Integration

With the multitude of connected devices and platforms available on the market, it's important to ensure that your software is compatible with a range of operating systems (iOS, Android, Windows etc).

Ensuring compatibility across platforms allows a wider range of users to access and interact with your connected device - regardless of their preferred platform. This is beneficial as it allows for a broader user base and maximized market potential.

In addition to compatibility with operating systems, integration with third-party platforms and services is also crucial for creating a comprehensive ecosystem around your connected device.By integrating with popular third-party platforms, such as cloud storage or IoT platforms, you enhance the functionality and interoperability of your device.

As an example, integrating your software with Cloud storage offers benefits such as data backup, synchronization, and remote access - and allows users to store their data in a centralized location, which ensures data integrity and accessibility from anywhere. This integration also allows for data synchronization across multiple devices, ensuring a consistent user experience across platforms.

Integration with third-party services, such as payment gateways or social media platforms, can add additional value to your connected product. These integrations provide users with a range of valuable functionality, such as providing the means to make secure transactions directly from your device, or allowing them to share updates and interact with their social networks seamlessly. By incorporating these integrations, the connected software expands its capabilities and provides a more comprehensive and user-friendly experience.

2. Security and Privacy

When building software for a connected device, ensuring the security and privacy of user data isa paramount concern. **The following are some key considerations for building secure and privacy-focused connected software.**

Data encryption and secure communication protocols:

One of the fundamental pillars of security in connected software is data encryption. To keep users' data safe, it is crucial that you implement robust encryption mechanisms, such as industry-standard algorithms that encode sensitive data and render it unreadable to unauthorized parties.

Employing secure communication protocols, such asSSL/TLS is also critical to security, as it helps ensure that data transmitted between a connected device and other endpoints remains confidential and protected from interception.

Secure user authentication and access controls:

To prevent unauthorized access to connected devices and the data they store, implementing secure user authentication mechanisms is essential. Strong authentication methods, such as multi-factor authentication, should be employed to verify the identity of users.

Access controls additionally bolster security by assigning appropriate privileges based on user roles, ensuring that sensitive functionalities and data are only accessible to authorized individuals.

Protection against vulnerabilities and potential cyber threats:

Connected software should be developed with a proactive security mindset to mitigate vulnerabilities and protect against potential cyber threats. It’s important to ensure that your IoT software development team utilizes regular security assessments and code reviews to help identify and address any security loopholes or weaknesses.

Following secure coding practices, adhering to industry security standards, and staying informed about emerging threats are key measures to safeguard against cyber attacks, data breaches, and unauthorized access.

Compliance with relevant privacy regulations:

In an era of heightened privacy concerns, compliance with privacy regulations is crucial.Regulations such as the CCPA (California Consumer Privacy Act) set stringent standards for the collection, processing, and storage of user data - and connected software must be designed to align with these regulations which ensure transparent data usage, proper consent mechanisms, and user rights.

Conducting privacy impact assessments, implementing data anonymization techniques, and providing users with control over their data are crucial steps in maintaining compliance and protecting user privacy.

3. Scalability and Performance

As connected devices gain popularity, the user base can grow rapidly - so connected software should be designed with scalability (the software's ability to handle increasing user demand and growing concurrent connections) in mind in order to accommodate this expansion.

It’s important that your IoT software development team ensures that the software they build is capable of handling an increasing number of users without compromising performance. This can be achieved through scalable software architecture, such as distributed systems or cloud-based solutions, which allow for efficient resource allocation, load balancing, and horizontal scaling to ensure a smooth user experience, regardless of user growth.

Another crucial consideration when building connected software is efficient resource management, which is paramount for ensuring optimal performance. By employing resource management techniques, such as caching, compression, and data optimization, your development team can maximize the utilization of available resources, reduce latency, and improve responsiveness, ensuring that your software consistently delivers high-performance experiences.

Your development team will also need to ensure that your connected software can handle real-world scenarios - so load testing is an important consideration. Load testing involves subjecting your software to simulated high-volume usage scenarios, which allows your team to evaluate the software’s performance under stress.

By analyzing the software's behavior and performance metrics during load testing, developers are able to identify potential bottlenecks, optimize performance, and fine-tune resource allocation, ensuring it can handle peak usage periods.

4. Device Compatibility and Variability

Connected software should be designed to support a wide range of compatibility and variability, including the ability to support various types of devices, and seamlessly integrate with various firmware versions.

Supporting various device types and models

The world of connected devices is evolving at a rapid pace, resulting in frequent hardware updates and a near constant introduction of newer device models. This continuous growth means that consumers often expect to have the ability to utilize multiple devices for their purposes.

For these reasons, connected software must be designed to support a wide range of device types and models. Whether it's smartphones, tablets, wearables, or smart home devices, compatibility across diverse devices is essential - and your development team should ensure that your software is able to function effectively across different devices and screen sizes, as well as the ability to maintain constant functionality and usability across various user interfaces.

Differences in hardware capabilities and specifications

Not all devices are created equal. When it comes to various types of devices, such as phones, tablets, laptops etc, there is often quite a bit of variation when it comes to their hardware capabilities (various sensors, GPS, cameras etc).

If you are planning to build your software in a way that allows it to be accessed by a variety of device types, it’s crucial that your IoT software development team accounts for those differences while building your software. If your software doesn't perform well on a particular device due to its limited hardware capabilities, users will have a frustrating experience, leading to negative user experiences.

Additionally, you'll want to ensure that your connected software is designed to seamlessly integrate with firmware updates and new firmware versions.

In addition to firmware versions, you'll also want to ensure backward compatibility with older device models. Accommodating these older devices ensures that users with legacy devices will still have the ability to benefit from the software's functionality.

Graceful degradation is another important aspect to consider, as this allows the software to adjust its features or performance based on the capabilities of the device, guaranteeing a consistent and positive user experience across a broad range of devices, regardless of their age or specifications.

To ensure your software is fully accessible across devices, you'll want to make sure your IoT software development team conducts extensive testing across many devices, hardware configurations, and firmware versions, including functional testing, usability testing, and compatibility testing

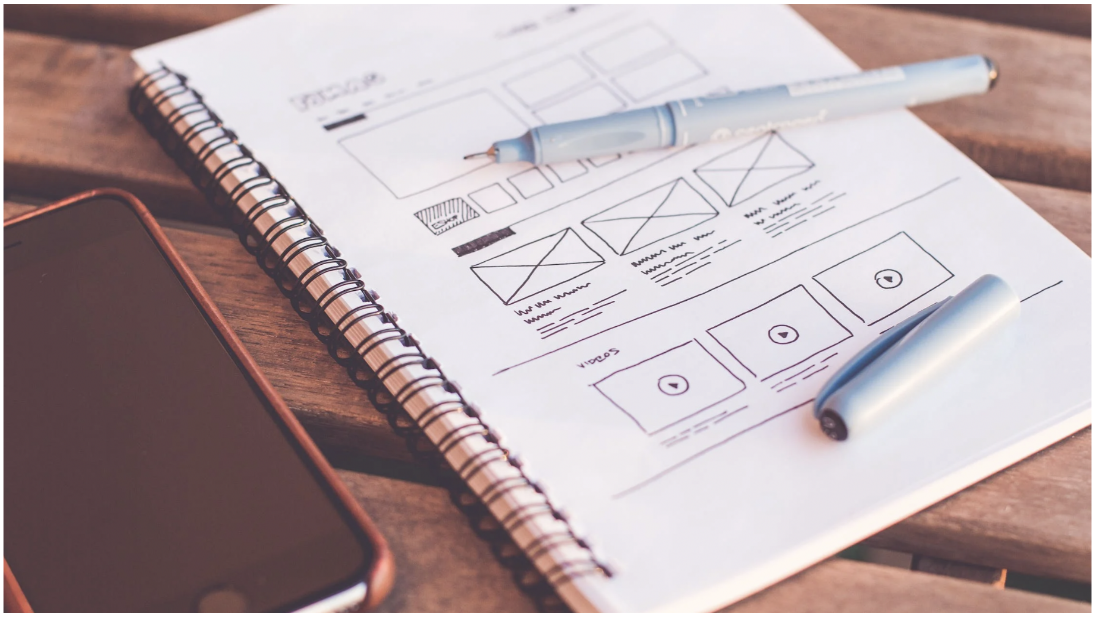
5. User Experience and Interface

When building connected software, taking the time to facilitate exceptional UX design (user experience design) is critical to success. UX design is the process of creating meaningful, user-centered, intuitive, and enjoyable software experiences.

This is a very specialized process that entails research, user interviews, and workshopping that allows designers to fully understand who their user is, and how to anticipate their wants and needs.

In addition to UX design, you'll also want to spend a significant amount of time on your softwares UI (user interface) which encompasses the visual and interactive elements of your software. UI design is concerned with ensuring that users are able to easily and quickly navigate and interact with your app.

Another important aspect of design is ADA compliance - a complex system that outlines how to build digital products that allow for an equal experience for all people, taking into consideration users with both permanent and/or temporary physical or mental disabilities.This involves providing accommodations such as alternative text for images, keyboard navigation, adjustable font sizes, voice recognition technology, screen readers and more.



Because accessibility is so important, we've put together this [free checklist and guide that provides all the information necessary to understanding ADA compliance.]

The following are some additional considerations for designing impactful, user friendly software

- ## Create an Intuitive and User-Friendly Interface
  Your software should provide users with an intuitive and user-friendly interface that allows them to immediately understand how to navigate your software, interact with elements, and accomplish tasks without confusion or frustration.

- ## Make it Efficient
  Good UX design focuses on streamlining user workflows and processes, enabling users to achieve their goals efficiently. When designing your connected software, ensure that tasks and actions are optimized for speed and ease of use.

- ### Make it Consistent

  Ensure that design elements, layouts, and interactions are consistent throughout your software. This consistency helps users build mental models and confidently predict how different parts of the product will work - allowing for more efficiency and far less frustration.

- ### Provide Clear Information Hierarchy

  A clear information hierarchy in UX design refers to the organization and presentation of information in a way that helps users quickly understand the importance and relationship of different elements. This plays a crucial role in guiding users through the product and ensuring they can easily find what they need.

  Users should be able to identify the most crucial information or actions easily, information should be grouped logically into categories or sections, and a clear visual hierarchy should be utilized through design elements such as size, color, contrast, and typography.

- ### Utilize Responsive Design:

  Different devices have different screen sizes, so it’s important that designers ensure that the software's user interface is responsive and adaptable to various device form factors. Responsive design ensures that your interface will remain usable and visually appealing across devices by dynamically adjusting based on the device type.

- ### Branding and Visual Identity

  Consistency with your brand's visual identity is essential for establishing a strong and recognizable brand presence. To achieve this, designers should incorporate your brand's colors, typography, and visual elements consistently throughout your software's UI.

6. Connectivity and Network Reliability

Considering our increasing reliance on connected devices, it's understandable that ensuring your software is accessible and functional whenever and wherever a user decides to utilize it is incredibly important.

To maximize your softwares reach and accessibility, and to help ensure that users are able to connect their devices regardless of the network technology, it's important that your connected software is designed to support various network protocols, such as Wi-Fi, Bluetooth, and cellular connections. By doing so, you help ensure that your software has the ability to leverage the available network infrastructure - providing users with consistent and seamless service.

However, as we're all aware, intermittent connectivity issues can be a common occurrence. For this reason, your connected software should be designed to handle connectivity issues gracefully, without causing disruption to the user experience. This can be done by implementing techniques such as intelligent buffering, offline caching, and asynchronous data synchronization, which allow the software to provide smooth operation even during periods of limited or no connectivity.

For instances when users temporarily have no connectivity, offline functionalities and data synchronization mechanisms can be a game changer. By incorporating offline functionality into your software, you allow users to continue using certain features even when not connected to a network - for example, a note-taking app may allow users to create or edit notes offline, and then provide automatic synchronization once a connection is again available.

As per the previous example, providing data synchronization ensures that any changes a user makes offline are seamlessly updated across devices and platforms once it reconnects with a network. This keeps the user's data consistent and up to date, regardless of the device they use.For instance, a cloud storage app can automatically synchronize files across devices, ensuring that the latest version of a file is always accessible.

Finally, it’s important to ensure that your connected software monitors and optimizes network usage, as doing so allows your app to provide efficient and responsible utilization of network resources. Developers can help reduce strain on network infrastructure and enhance overall performance by creating systems that manage network bandwidth effectively, for example by minimizing unnecessary data transfers and implementing efficient data compression techniques.

7. Data Management and Analytics

Connected devices generate massive amounts of valuable data, making effective data management and analytics crucial. Implementing a robust data analytics system unlocks the true potential of this data and offers numerous benefits for both users and development teams.

Data analytics enable development teams to gain actionable insights into user behavior, preferences, and usage patterns, which can be leveraged to enhance the user experience. For example, by analyzing user data, a smart home system can learn the occupants' daily routines and automatically adjust lighting, temperature, and other settings to align with their preferences, creating a more personalized and comfortable living environment.

Furthermore, data analytics can empower users by providing personalized recommendations and insights. For instance, a connected health and wellness app can analyze user data, such as exercise habits and sleep patterns, and then offer tailored suggestions for improving overall well-being. This could include customized exercise plans, sleep optimization techniques, and nutritional recommendations based on individual needs and goals.

Data analytics also enables predictive capabilities that can enhance the user experience. By leveraging machine learning algorithms, a smart assistant can analyze user interactions and preferences to anticipate their needs and proactively provide relevant information or assistance. For example, a voice-activated assistant can learn a user's preferences for music, news, and weather updates and provide personalized recommendations without the need for explicit requests.

Another benefit of data analytics is anomaly detection. By monitoring and analyzing data from connected devices, potential issues or abnormalities can be identified in real-time. For instance, an energy monitoring system can analyze electricity usage patterns and detect unusual spikes or irregularities, which can indicate malfunctioning appliances or potential safety hazards. This enables timely intervention and preventive maintenance, ensuring a safer and more efficient user experience.

Moreover, data analytics can contribute to product improvement and innovation. By aggregating and analyzing data from a large user base, development teams can identify usage patterns, emerging trends, and areas for improvement. This information can guide the development of new features, enhancements, and even entirely new products that better meet user needs. For example, a smart home security system provider can analyze user data to identify common vulnerabilities or user concerns and develop enhanced security features to address them.

Data analytics can go beyond improving the user experience and enable proactive device management. By leveraging predictive analytics techniques, connected device software can anticipate device failures, identify maintenance requirements, and optimize device performance. For example, by analyzing sensor data, a smart thermostat can proactively detect anomalies, such as temperature fluctuations, and trigger maintenance notifications or automatically adjust settings to prevent system failures.

8. Battery Efficiency and Power Optimization

When building connected software, optimizing battery efficiency and power consumption is essential to providing a positive user experience. By implementing strategies to minimize power consumption and efficiently handle background processes and notifications, developers can ensure that the software operates efficiently and conserves battery power.

One key consideration is optimizing the software itself to minimize power consumption. This involves identifying and optimizing resource-intensive tasks, such as excessive CPU usage, unnecessary network requests, or continuous location tracking. By reducing these activities, the software can minimize the strain on the device's battery.

For example, a navigation app can employ smart algorithms to minimize GPS usage and optimize route calculations, resulting in reduced battery drain.

Efficient handling of background processes and notifications is another crucial aspect of power optimization. Background processes, such as data synchronization or updates, should be carefully managed to prevent excessive battery usage.

Implementing intelligent scheduling mechanisms can ensure that background tasks are executed when the device is connected to a power source or during periods of low user

activity, thus minimizing their impact on battery life. Similarly, managing notifications and their frequency helps prevent unnecessary wake-ups and reduces battery drain caused by constant display or network activity.

Implementing power-saving modes or features within the software is another effective strategy.Power-saving modes allow users to customize settings that optimize power consumption, such as reducing screen brightness, disabling background data usage, or limiting the functionality of non-essential features. By providing these options, users can choose the level of power optimization that suits their needs and extend their device's battery life.

Efficient use of sensors and connectivity modules is crucial for battery optimization. Connected devices often rely on various sensors, such as accelerometers, gyroscopes, or ambient light sensors, which consume significant power. By intelligently utilizing these sensors only when necessary, the software can minimize power drain. Additionally, optimizing the use of connectivity modules like Wi-Fi, Bluetooth, or cellular data by employing efficient algorithms and techniques reduces unnecessary power consumption.

Continuous monitoring and optimization of power usage is vital for maintaining battery efficiency. Developers should regularly analyze power consumption patterns and identify areas for improvement. By using profiling tools and analyzing energy usage reports, they can pinpoint areas of the software that are consuming excessive power and make the necessary optimizations. This iterative approach ensures that power optimization efforts remain effective over time and with software updates.

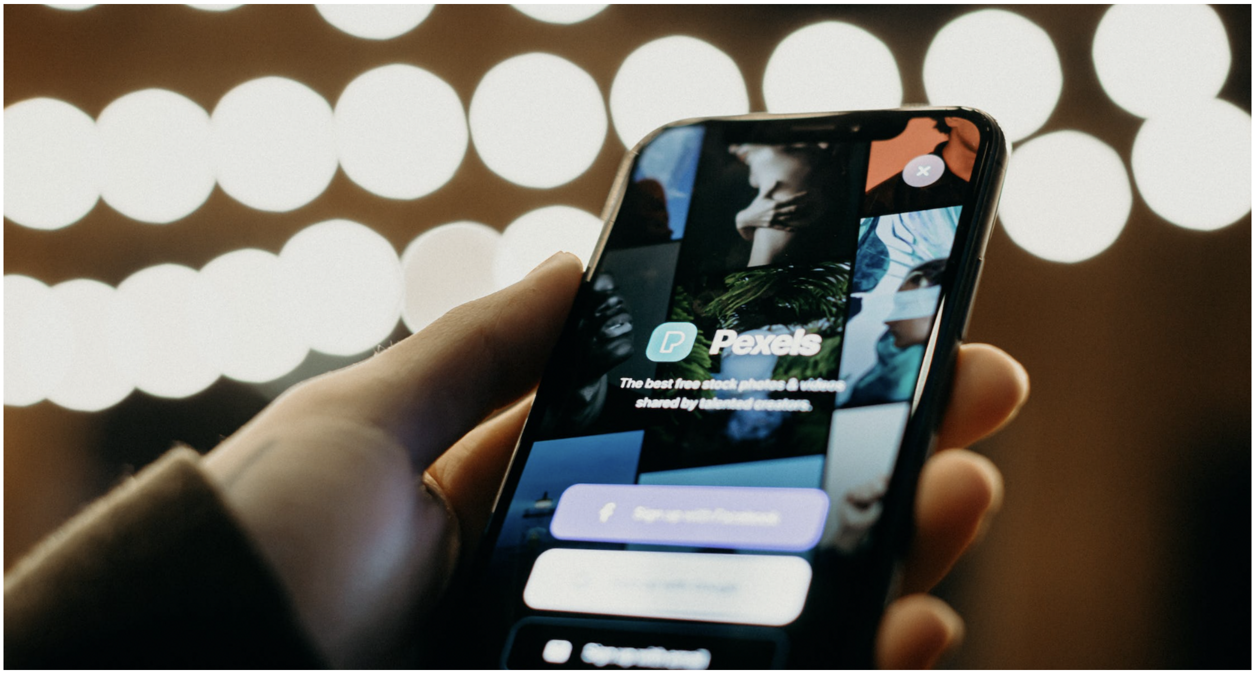
BUILDING AN IOT APP THAT USERS LOVE: OUR TOP 5 TIPS

Survival of the fittest doesn't just apply to the animal kingdom anymore. In fact, it's become the mantra of today's digital landscape, where fresh and exciting products go to market daily - each of which must desperately vie for the attention of users.

In this incredibly dynamic environment, it’s your connected product's ability to hook users from the get-go that is the ultimate test of its survival - and your success.

If your connected product is to survive and thrive in this fast-paced digital world (and not become one of the 21% of apps that are abandoned after just one use) you’ll need to create an experience that immediately hooks users - and lures them back day after day. Of course this is easier said than done - but never fear, there *is* a formula for building connected software that people love to use.

In this section, we'll explore our top five creative strategies for designing and developing a connected app that not only captures attention but also motivates users to keep coming back for more.


Pexels
The best free stock photos & videos shared by talented creators.

1. Simplify Onboarding with a Seamless Experience:

First impressions matter - A LOT - so when it comes to app onboarding, a seamless experience can make all the difference.

To help ensure users understand your app's features and benefits from the get go, it’s crucial that you create an onboarding process that quickly and easily guides them through your app's initial setup. Allowing a user to become frustrated during their first engagement with your product is a death sentence for continued engagement.

By streamlining the onboarding process, you eliminate barriers to entry and increase the chances of user retention - so take inspiration from the simplicity of apps like Instagram, where users are prompted to capture and share photos within seconds of signing up.

For an in depth exploration on the various ways you can create a seamless onboarding experience, take a look at our blog article, Mastering the Art of Seamless Onboarding for Connected Software.

2. Create a Gamified Experience:

Creating a gamified experience is a powerful strategy to enhance user engagement in connected software, as it taps into humans' innate love for games and competition, making it a game-changer for keeping users hooked. By integrating gamification elements into your connected app, you tap into users' intrinsic motivation and create an engaging experience that keeps them coming back for more.

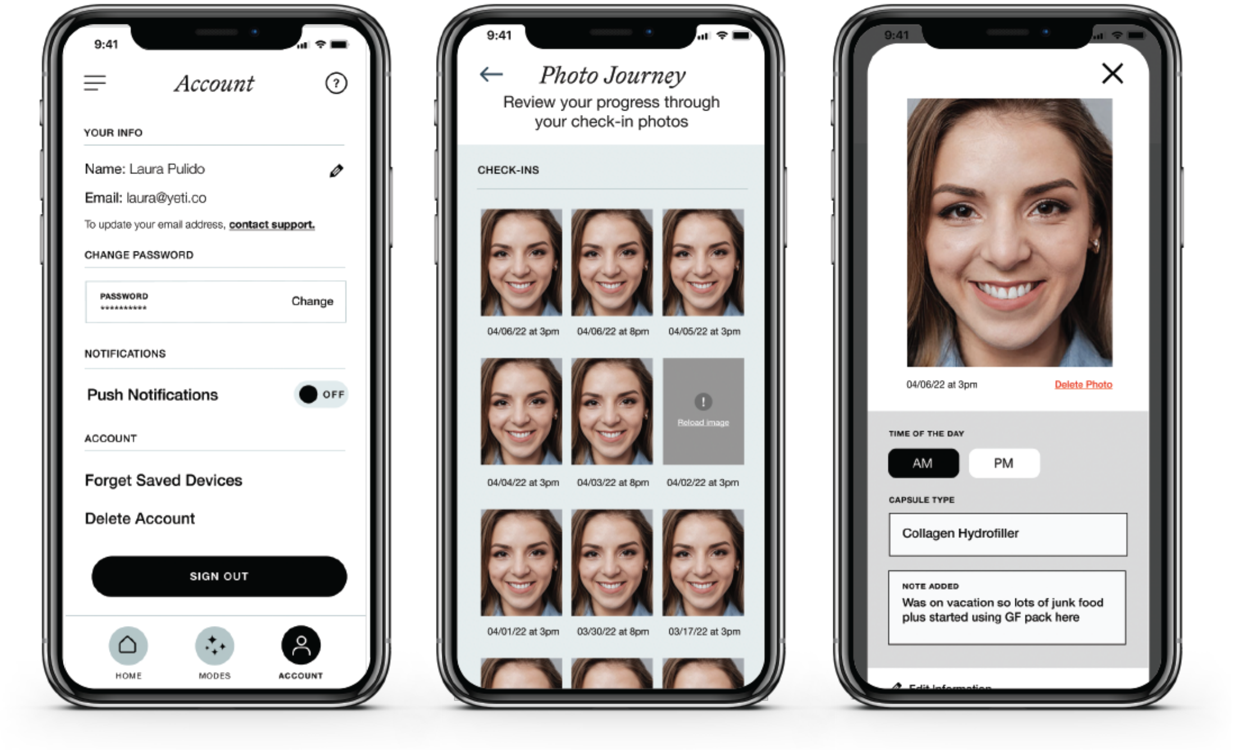
To engage users and enhance their experience, gamification can be utilized in a multitude of ways. One common method is the use of points, badges, or levels to represent users' progress and achievements. By earning points or unlocking badges, users feel a sense of accomplishment and are motivated to continue using the app.

Leaderboards are another popular gamification element that allows users to compare their performance with others, fueling friendly competition and a desire to climb to the top. Challenges and quests add an element of excitement and goal-oriented gameplay. Rewards, whether virtual or tangible, can also be a serious motivator for users, and can take many forms: discounts, access to premium features or exclusive content and many more.

By incorporating these common gamification elements, apps create a captivating and interactive experience that keeps users hooked and motivated to explore and engage with the app's functionalities.

3. Personalize and Tailor the User Experience:

Personalizing user experience is a highly effective strategy for enhancing engagement in connected software, as users appreciate when an app understands their individual preferences and delivers content that aligns with their specific needs. By leveraging user data and behavioral patterns, you can create personalized experiences that cater to each user's unique interests, ensuring a higher level of engagement and satisfaction.



Name: Laura Pulido

Email: laura@yeti.co

To update your email address, contact support.

PASSWORD **********

Push Notifications OFF

Review your progress through your check-in photos

04/06/22 at 3pm

TIME OF THE DAY: AM

CAPSULE TYPE: Collagen Hydrofiller

NOTE ADDED

Was on vacation so lots of junk food plus started using GF pack here

Connected software provides ample opportunities for personalization. For example, a smart home automation app can learn users' routines and preferences over time, automatically adjusting temperature settings, lighting preferences, or device behavior to match their preferences. This personalized approach enhances the user experience by seamlessly integrating with users' lifestyles and eliminating the need for manual adjustments.

Spotify, the popular music streaming app, is another excellent example of personalization in connected software. The app analyzes users' listening habits, favorite genres, and artist preferences to curate custom playlists and offer personalized recommendations. By tailoring the music experience to each user's taste, Spotify keeps users engaged, delivers content they genuinely enjoy, and fosters a sense of connection with the app.

Incorporating personalization can extend beyond content recommendations. One example might include personalizing notifications, alerts, or reminders based on user preferences and behavior. For instance, a health and fitness app can send personalized reminders for workout sessions or provide tailored insights based on users' fitness goals and progress. By delivering relevant and timely information, the app becomes an indispensable tool for users, keeping them engaged and motivated.

4. Foster Community and Social Interaction:

Humans are social creatures, so fostering community and social interaction within your app is a powerful way to boost user engagement. This can be achieved by implementing various features that facilitate interaction and collaboration among users, creating an environment where they can engage with one another, share experiences, and feel a sense of belonging.

Connected software offers numerous opportunities to incorporate community-building elements. For example, a fitness tracking app can include features that allow users to join fitness challenges, create virtual workout groups, or share their achievements with others. By providing a platform for users to connect, support, and motivate each other, the app becomes more than just a solitary fitness tool—it becomes a social space where users can find inspiration and encouragement.

To foster social interaction, you'll want to include a few basic features for your connected software including: chat functionalities, discussion boards, or user forums where users can connect, ask questions, and provide feedback. Encourage users to share their experiences, achievements, or challenges within the app or on social media platforms. Creating a space where users feel heard, supported, and valued fosters a sense of community and cultivates a loyal user base. Gamification can also seamlessly intertwine with the social features of an app, creating a more dynamic and engaging user experience.

5. Continuously Improve and Innovate:

Technology evolves rapidly, user expectations change, and new trends emerge - so continuously improving your connected app is a crucial aspect of maintaining user engagement and staying ahead in a competitive market. To keep users hooked and interested, it's essential to consistently enhance your app's features, functionalities, and overall user experience.

Regularly gathering user feedback is a valuable practice that will allow you to understand user needs, preferences, and pain points - so you'll want to implement feedback mechanisms such as surveys, ratings, or user reviews that will allow you to gather insights directly from your user base. Analyzing this feedback will allow you to identify areas for improvement and prioritize enhancements accordingly. By listening to your users and addressing their concerns or suggestions, you demonstrate a commitment to their satisfaction and create a user-centric app that evolves based on their needs.

Because innovation is a key driver of user engagement you'll need to stay up-to-date with the latest industry trends, emerging technologies, and user behaviors. This will allow you to Identify opportunities for introducing new features or integrations - this could involve leveraging advancements in ai, vr, or Internet of Things (IoT) to enhance your app's capabilities and create novel user experiences. By offering something new and exciting, you captivate users' interest and motivate them to continue using your app.

Creating a connected app that captivates users and keeps them coming back requires a blend of creativity, usability, and personalization. So, put these strategies into action, and get ready to create an app that becomes an integral part of users' lives!

5 COMMON PITFALLS IN IOT SOFTWARE DEVELOPMENT

Building software for connected devices requires careful consideration and attention to detail. There are several common pitfalls that developers are prone to falling into - mistakes that can negatively impact the app's functionality, security, and user experience.

The following are the top five mistakes we see made in the IoT software development process and the ways you can avoid them.

1. Not Focusing on the User's Experience

New connected devices appear on the market daily - some are successful.. but many will fail to thrive. What are the differentiating factors in these cases?

The truth is that creating successful connected software isn't just about building something that "works" - it's about creating software that works well and feels good to use - which is generally a result of great **UX (user experience) design**.

UX design is all about optimizing a users' experience by understanding their needs, and creating a connected app that will allow them to meet those needs in the easiest and most pleasurable way possible.

A good test for determining whether an app has good UX design is how long it takes a new user to learn how to use the app proficiently - if they can start using the app quickly, without much guidance, then the UX has done its job.

Because good UX design is nearly invisible (people usually only notice UX when it creates a difficult experience), it's common for people unfamiliar with the app development process to have a hard time understanding why they should spend the time and money on UX Design.

Aside from creating a positive experience for your user, spending time on UX Design:

- **Ensures your product will attract and retain users**
- **Creates more revenue. Users are more likely to pay for something that meets their needs and is easy to use - and they'll be more likely to recommend your product**
- **Gives you room to grow. UX design is all about knowing who your users are, helping you to anticipate and build for future needs.**

When creating your connected software, be sure that your design team isn't skimping on UX design - and that they are well-versed in UX design principles. Remember, the success of your connected devices hinges on the user experience you provide. A well-designed and intuitive interface can make the difference between a thriving product and one that falls short of expectations.

2. Lack of Compatibility Testing:

Failing to conduct thorough **compatibility testing** is a common mistake that can have detrimental effects on the user experience and the overall functionality of connected software. Compatibility issues often arise when software is not adequately tested across various devices, operating systems, and firmware versions. Without this comprehensive testing, users are likely to encounter poor experiences, such as glitches, crashes, or features that don't work as intended.

To avoid compatibility issues, it is crucial that your development team allocates sufficient time and resources for comprehensive compatibility testing. This process involves testing the software on a wide range of devices, including different models, screen sizes, and hardware capabilities. Additionally, testing across various operating systems, such as iOS, Android, or Windows, is essential to ensure seamless integration and functionality across platforms.

Compatibility testing should not be limited to just the latest versions of operating systems or firmware. It is equally important to test compatibility with older versions, as some users may still be using them. By including backward compatibility in the testing process, you can ensure that your software functions properly across a wide range of devices and versions, providing a consistent experience for all users.

3. Overlooking Scalability and Performance

As the popularity of your connected device grows, so will the demand for your software. Unfortunately, if your development team has failed to consider scalability and has built software incapable of handling this increased user demand, the result will be sluggish performance and frustrated users.

To ensure users always have an incredible experience with your product, make sure that your development team is designing your software architecture with scalability in mind. This will involve creating a flexible and modular architecture that can easily accommodate additional users and data. By anticipating future growth and designing your software to scale horizontally or vertically, you can avoid bottlenecks and ensure that your software can handle increasing user demand seamlessly.

Failing to **load test** is a similar mistake that developers can make. Load testing simulates realistic usage scenarios and allows you to measure the software's response time, resource utilization, and overall stability under heavy loads. This valuable information empowers you to make necessary adjustments and optimizations before your software faces a surge in user traffic and critical issues occur.

The benefits of load testing extend beyond immediate fixes. Because it provides you with an understanding of your software's limits and performance thresholds, you can proactively invest in the necessary infrastructure, resource allocation, and optimizations, which will allow you to accommodate growing user demand. Load testing acts as a safeguard, allowing you to address potential issues early on and ensure a smooth user experience as your product gains traction.

4. Failing to User Test

Developing an app is time consuming and costly, so before diving headlong into the process it's important to make sure that the product you're creating is one that people actually want and need.

Far too often we hear about apps that were created with the assumption that the features and design would be a huge success with users...only to be met with crickets upon launch.

User testing is the process of understanding the user's experience of your app, an app feature, or even your idea for an app. At the most basic level, it's about testing and quantifying how someone uses or thinks about your product — which is often different than how you think they should.

User testing allows you, the product creator, to gain a sense of how real people will interact with your product. You can see what pitfalls they might encounter, their reaction to the layout and the colors - and you can determine if they view your product as something useful that they would actually use.

Continuously testing throughout the development of your app will allow you to create an intuitive product that truly provides value to its users - and it will provide you with the ability to move forward with ideas, designs and features that work on every level, while avoiding countless hours on those that don't match your users expectations.

5. Overlooking Compliance

Ensuring full compliance with applicable laws, regulations, and industry standards is of utmost importance when developing software in the United States. Neglecting to dedicate the necessary time and effort to ensure compliance can have severe consequences, including the imposition of fines, lawsuits, legal actions, and more **Before launching your software, it's crucial that you ensure it's compliant with all of the following:**

Data Protection and Privacy

If your software handles personal data, it must comply with relevant data protection and privacy laws, such as the **California Consumer Privacy Act (CCPA)** and the **General**

Data Protection Regulation (GDPR). Compliance involves obtaining user consent for data collection, implementing proper data handling practices, providing data access and deletion mechanisms, and maintaining data security. For example, a social media platform must comply with privacy regulations by allowing users to control their data sharing preferences and providing mechanisms to delete their data upon request.

Intellectual Property Rights:

Software must respect intellectual property rights, including copyrights, trademarks, and patents. It should not infringe on the proprietary rights of others. Developers need to ensure they have the necessary licenses and permissions to use third-party software libraries, frameworks, or APIs. For instance, a video streaming service must properly license and attribute the content it streams to avoid copyright violations.

ADA Accessibility

Software should comply with accessibility standards, such as the Web Content Accessibility Guidelines (WCAG), to ensure equal access for individuals with disabilities. This involves designing interfaces that are perceivable, operable, understandable, and robust for users with visual, auditory, or motor impairments.

For example, an e- commerce platform should provide alternative text for images, captioning for videos, and keyboard navigation options to accommodate users with different abilities.

To learn more about Accessibility compliance, download our free ADA Accessibility Checklist and Guide.

Security and Cybersecurity

Software should adhere to industry best practices and security standards to protect against cyber threats and data breaches. This involves implementing secure coding practices, regularly updating software with security patches, conducting vulnerability assessments, and employing encryption and authentication mechanisms.

For example, a banking application must comply with Payment Card Industry Data Security Standard (PCI DSS) requirements to safeguard customer financial data during transactions.
Export Controls

Software may need to comply with export control laws and regulations, such as the International Trac in Arms Regulations (ITAR) or the Export Administration Regulations (EAR). This is particularly important for software that contains encryption algorithms or technology with potential military or strategic applications. Compliance involves obtaining necessary licenses and ensuring that the software is not exported to prohibited destinations or unauthorized end-users.

We hope you've found Unlocking Connectivity: The Ultimate Guide to IoT Software Development to be a helpful resource for beginning your IoT journey. For more helpful resources, be sure to check out **our "Learn" page** and **the Yeti blog.**

If you're starting an IoT project, and could use the help of an experienced team, we'd love to chat! Feel free to reach out with any questions you may have and we'll get back to you in two shakes of a Yeti's tail!