Product Requirements Document (PRD)

Please remove help/template/example texts and tips from your submission (everything but headers).

*The goal of the PRD is to address four critical risks:*

| **Risk** | **Perspective** |
| --- | --- |
| Value | User |
| Usability | User |
| Feasibility | Engineering |
| Viability | Business |

*The PRD helps align the development team, stakeholders, and other involved parties regarding the product's vision and scope. The PRD is the main document that will help manage expectations and get everyone on the same page. Make sure to ask your stakeholders to* ***explain*** *their understanding of key sections of this document after reading it.*

We are also including an imaginary, simplified, **Spotify** case study.

[Problem Description](#_m14ebq69irye)

[Scope](#_h095bb6e23m6)

[Use Cases](#_a494vmyn4i0c)

[Purpose and Vision (Background)](#_a1rehjp14t6a)

[Stakeholders](#_ni1nsrov0hlq)

[Preliminary Context](#_8fsnkm7d5o2g)

[Assumptions](#_o6z1frxke2z)

[Constraints](#_q1it1bwhygi3)

[Dependencies](#_hn9mh6b6m132)

[Market Assessment and Competition Analysis](#_kztlc8doxc1v)

[Target Demographics (User Persona)](#_w2boxoybxhab)

[Requirements](#_agmoxh3nwlca)

[User Stories and Features (Functional Requirements)](#_26r2y8q9o6bs)

[Non-Functional Requirements](#_phuqv9aimzm2)

[Data Requirements](#_89injk5wussz)

[Integration Requirements](#_ti0b4uln9ysy)

[User Interaction and Design](#_jq7dpkh2hu4j)

[Milestones and Timeline](#_u3bp37hkrgxj)

[Goals and Success Metrics](#_ybblbzrnua5i)

[Open Questions](#_9p4b6jpz1kep)

[Out of Scope](#_c0bknaj5akzz)

# **Problem Description**

*Describe the problem from a user perspective. Do not describe the solution to the problem, even if someone already thought of the solution.*

“The framing of a problem is often far more essential than its solution.” ― **Albert Einstein**

Traditional music consumption methods lack personalization and ease of access. It’s inflexible, impractical, or expensive. Users struggle to discover new music tailored to their tastes.

**TIP** People have problems. Consider and write about the problem from all stakeholders’ points of view. Write about a paragraph for each point of view describing what it is they’d like to address using technology. For example, a researcher might be looking for a better way to access and clean data. Or a business might want to increase their customer base by adding a feature to their business web presence.

## **Scope**

*Define the preliminary scope of the problem you want to tackle. You might only focus on a subset of the problem, delivering a proof-of-concept, or refactoring existing code. There’s an Out of Scope section later in the PRD.*

The scope for this product will be limited to searching and streaming songs and albums. Stretch goals include: liking songs, recommending similar songs, managing the queue of tracks coming up, creating and discovering playlists, downloading songs for offline listening, ...

We’ll limit our scope to the US market because of streaming constraints and rights dependencies listed below.

**TIP** Problem scope is a list of all the parts of the problem you think you can address. When starting a project, many software developers overpromise what can be achieved. Give yourself time and space for research and learning, in addition to development. Remember, you can always revise the scope and add *stretch goals* later.

## **Use Cases**

*List concrete use cases that would benefit from a solution to that problem. You’ll use these cases to test your solution. It should be driven by the scope, but you can list more.*

We want to test the searching and streaming services first.

The user will search for a specific track by song name or artist, then listen to that track. The user will play and pause the track at their leisure.

The user will search for a specific album by name or artist, then listen to the whole album. The user will want to skip songs to come back to another one.

# **Purpose and Vision (Background)**

*Describe the purpose or vision from the perspective of the stakeholders, business, research lab, or technology. Most entities will tackle problems that align with their short to long-term goals. Describe the entity's purpose, and how this project or product fits their strategy.*

*The most successful companies are driven by a strong vision, and all employees help each other towards achieving that vision.*

Our purpose is to develop a music streaming platform that provides users with an exceptional music listening experience, personalized recommendations, and a seamless user interface.

We want to become the de-facto platform for listening to any music while also letting artists get their money for the music they produce.

Up to now, listening to music is either: very expensive (iTunes, CDS, pay per song or pay per album, audio setup), inflexible (you buy the whole album, you listen to the radio), or impractical (you have to go to the shop or need to know what you want to listen to or buy).

# **Stakeholders**

*List people that have an interest in the project:*

* *How often do they need to be updated?*
* *What type of updates do they require?*
* *Are they decision-makers?*
* *If yes, which decisions should go through them?*
* *What do they require to make an informed decision?*

*Decision-makers should request more information to be able to decide. Uninformed decision makers will make bad decisions for you and your team. Be wary of stakeholders that go with instinct, validate their claims and be ready to challenge them with data. Not everyone has the deep-instinct that comes with long industry experience.*

* Seed Investors
* CEO/CTO/Founders
* Engineering Managers
* Product Managers
* Engineering Team
* Marketing Team
* Legal Team
* Sales
* Users

# **Preliminary Context**

*Before you deep dive into user persona, requirements, and timeline, list what you already know or expect. This will kickstart the discussion about a potential solution to the problem.*

“If a problem can't be solved within the frame it was conceived, the solution lies in reframing the problem.” ― **Brian McGreevy,** [**Hemlock Grove**](https://www.goodreads.com/work/quotes/17496592)

## **Assumptions**

*Assumptions are factors or conditions that are believed to be true but haven't been fully validated yet. It's essential to identify and document these assumptions to ensure everyone involved in the project is aware of potential risks and uncertainties.*

*List any assumptions you have about your users, technical constraints, or business goals.*

*Examples:*

* *Target user's preferences and behavior*
* *Availability and reliability of third-party APIs or services*
* *Compatibility of the product with certain devices or platforms*
* *Project timeline and resource availability*

We can easily acquire the songs in the correct format and with the right metadata from labels.

We can develop an app that works across all platforms (iOS, Android, Windows, macOS, Linux) and divide sizes.

Libraries used for development are thoroughly tested and robust.

We have access to users that will provide valuable feedback in a timely manner (i.e., weekly, daily).

We have three months to prove core functionality, and one year until our seed funding runs out.

## **Constraints**

*Constraints are limitations and restrictions that may impact the development and implementation of the product. Understanding constraints helps manage expectations and identify potential challenges that need to be addressed during the project.*

*Examples:*

* *Budgetary constraints and financial limitations*
* *Technology constraints, such as specific hardware or software requirements*
* *Time constraints and project deadlines*
* *Regulatory or legal constraints that must be adhered to*
* *Resource constraints, including staffing and expertise limitations*

We need to make sure that artists and labels are properly compensated for each track listened to, this means that we have the rights to share the music.

Our network infrastructure must scale with the searches and streams; we can’t create too much latency for users.

We are a small team, and we should start small but functional. We want to test our core features in 3 months from now.

## **Dependencies**

*Dependencies are elements or tasks that rely on the completion of other tasks or external factors to move forward. Identifying dependencies is crucial for effective project planning and management to avoid bottlenecks and delays.*

*Examples:*

* *Dependencies on external vendors or suppliers to provide necessary components*
* *Software dependencies, where certain libraries or frameworks need to be integrated*
* *Data dependencies, such as relying on data from a separate system or database*
* *Sequential dependencies, where tasks need to be completed in a specific order*

We’re dependent on getting the rights to play songs. We should work with labels to get them, and negotiate a revenue stream for them and us.

If we have a freemium model, we could rely on ads (meaning we need to integrate with an ads network), or initial loss to get network effect and users.

We’re dependent on labels to provide us with the songs, tracks, albums, and corresponding metadata.

We need a database and datastore with tracks before being able to search and stream them.

Are we using any existing libraries to power our app? How reliable is that library?

# **Market Assessment and Competition Analysis**

*List existing alternatives to your product or project, including open-source projects and proprietary solutions. You could use methodologies such as SWOT, Porter’s Five Forces, PEST, Business Model Canvas, … to help structure your thoughts. There is no one size fits all approaches.*

*The goal of this section is to see whether we can re-use something that exists, get inspired by it, or position ourselves differently compared to the competition (see Blue vs Red Ocean Strategy).*

*For each alternative, explain* ***why it is or is not a good fit*** *for your problem. If it is a good fit, maybe you should use that, or improve it if it’s open-source.* ***We don’t want to reinvent the wheel.***

Alternatives:

* YouTube, DailyMotion: you also get the video, and content is not limited to music.
* Torrenting: it’s illegal and users require technical know-how, but it’s free, but artists don’t make money
* WinAmp and other media players: allow to play tracks but you need to have them at hands
* iTunes: expensive, pay-per-track.
* Radio: no flexibility, can’t go back to listen to a track, difficult to know what song is playing
* TV: similar to YouTube but with the inflexibility of radio

Playback capabilities already exist and can probably be re-used. [List of libraries doing that.]

Streaming capabilities also exist (i.e., YouTube) and we expect them to be lighter for our use cases (audio has a smaller footprint). [List of libraries/technologies doing that.]

# **Target Demographics (User Persona)**

*Define* [***user persona***](https://docs.google.com/document/d/1fq4ZdIeErZpQutTrKMYeknMd_B90oq5JwyVKNUCXLM8/edit?usp=sharing) *for your product. You probably don’t need more than one for the CS Capstone, but it’s worth listing all possible users and simply focusing on one.*

*For each user persona, think about the number of users that might be interested in this product.*

*Users can be businesses and their employees as well.*

*Every time that you think about a feature, put yourself in the shoes of your user persona. Does this bring value to them? Is it usable?*

There are different ways to look at this:

* Platform (mobile, desktop, listening devices)
* Paying vs ad-based (revenue model)
* Consumption style (YouTube, CDs, iTunes, radio, etc.)

We can craft a few persona that meet these criteria, e.g.,

* John is a 16 yo smartphone user that can’t pay for music tracks and is used to listening/watching clips on YouTube.
* Leslie is a 26 yo that loves to dance to all kinds of music, hence searches and downloads tracks illegally with torrenting and puts them on their phone for offline listening later.
* Sofia is a 54 yo that likes to listen to the radio in the car, and spends time listening to tracks at home. They have an extensive collection of CDs.

# **Requirements**

*The requirements will drive your software development. It will give you the big picture of what you’re building, and the whole team should be aware of and agree on it.*

## **User Stories and Features (Functional Requirements)**

*Specify the features and functionalities that the product should offer. Break down each feature into clear, detailed requirements (to be done in your project management tool, such as GitHub Issues).*

*User Stories:* ***“As a [persona = user], I [want to = intent], [so that = benefit].”***

* *Who are we building this for? We’re not just after a job title, we’re after the persona of the person. The team should have a shared understanding of who that user is. You've hopefully interviewed plenty of users. You understand how that person works, how they think and what they feel.*
* *What is it they’re actually trying to achieve? Describe the intent, not the feature.*
* *What’s the overall benefit they’re trying to achieve?*

*MoSCoW Prioritization:*

* ***M****ust Have: non-negotiable product needs that are mandatory for the team*
* ***S****hould Have: Important initiatives that are not vital, but add significant value*
* ***C****ould Have: Nice to have initiatives that will have a small impact if left out*
* ***W****ill Not Have: Initiatives that are not priority for this specific time frame*

*MoSCoW is only one of many frameworks to help you prioritize your requirements. It’s not a panacea. try being quantitative in prioritizing your requirements (standard set of cost and benefit criteria).*

| **User Story** | **Feature** | **Priority** | **GitHub Issue** | **Dependencies** |
| --- | --- | --- | --- | --- |
| As a student, I want to view my upcoming class schedule, so that I can plan my study and personal time effectively. | TBD | Must Have | TBD | N/A |
| As an online shopper, I want to filter search results by price range, so that I can quickly find products within my budget. | TBD | Should Have | TBD | N/A |

***After you have your first draft here, you can manage your backlog in the project management tool of your choice (GitHub recommended, using one issue per user story).***

## **Non-Functional Requirements**

*Define non-functional aspects such as performance, security, reliability, and scalability requirements. Include constraints related to budget, time, or technology.*

*Examples:*

* *Response time for user interactions should be less than 300 milliseconds.*
* *The system should support a 50% increase in user traffic during peak periods.*
* *The system should recover from a failure within 5 minutes without data loss.*
* *All communication between the client and server must be encrypted using SSL/TLS.*
* *The product should be available for access 24/7, except during scheduled maintenance windows.*
* *The product must be accessible to users with disabilities (compliance with WCAG guidelines).*
* *The product should work seamlessly on multiple devices and screen sizes (e.g., desktop, tablet, mobile).*
* *Code should be well-documented, following coding standards and best practices.*
* *The product should comply with relevant data protection regulations (e.g., GDPR, CCPA).*

***This will translate in specific tasks or put constraints on the functional requirements.***

## **Data Requirements**

*Outline the data that the product needs to collect, store, and manipulate. Define data formats and structures. Can live in a dedicated issue or document.*

## **Integration Requirements**

*Identify any third-party services, APIs, or systems that the product needs to integrate with. Describe the integration points and data exchanges. Can live in a dedicated issue or document.*

## **User Interaction and Design**

*Provide wireframes, mockups, or visual representations of the product's user interface. Describe the user experience design principles and interactions. Can live in a dedicated issue or document.*

# **Milestones and Timeline**

*Once you have a better idea of what you’re building and not building, it’s planning time. Initially, provide a ballpark estimate of what features will be delivered when, taking into account time, availability, skills, previous experience, and meetings/administrative overhead.*

*List the main features and place them in a timeline, including potential dependencies, and who might work on it. You can create milestones in GitHub and group issues in them. Issues can be assigned to individual contributors.*

*The milestones and timeline, like the rest of this document, is an* ***iterative*** *process. Your user stories will get more detailed as you approach implementation, hence you will have better estimates, and will be able to revise your timeline.*

# **Goals and Success Metrics**

*List project goals and the metrics you’ll use to judge its success. Success metrics help provide clarity, focus, and a means of evaluation.*

*Ensure that the goals and metrics are specific, measurable, attainable, relevant, and time-bound (SMART). Define a baseline and a target for each metric, and decide on the tracking method.*

*Avoid vanity metrics (e.g., the number of page views serves no purpose on a retail website if visitors don’t buy anything).*

E*xamples:*

| **Goal** | **Metric** | **Baseline** | **Target** | **Tracking Method** |
| --- | --- | --- | --- | --- |
| Increase user engagement | Average session duration | 1 min 34 sec | 2 min | Plausible Analytics |
| Product-market fit | How would you feel if you could no longer use this product? | Very disappointed < 40% | Very disappointed > 40% | Interview |

# **Open Questions**

*By this point, you probably have a lot of unanswered questions. Track them here, and leave some space for answers. Address these questions as you progress in time, and incorporate new knowledge in the other sections.*

# **Out of Scope**

*Identify elements that will not be included in the current project scope. Important for stakeholders to agree on expectations. You might only develop a proof-of-concept, not deploy it on target infrastructure, or not implement all security best-practices. You only have so much time.*