# どんなアプリを作るか「要件定義」する

## ①ポートフォリオの方向性　※AI音声でOK

こんにちは。いよいよこの講座も終盤に入りました。この動画ではweek9の全体像を説明します。

本日の流れはこちら。

💻本動画の流れ

（

・week8までの簡単な振り返り

・week9の全体像

・「未経験エンジニア転職」としてのポートフォリオとは

・技術要件をざっくり決める

）

まず、week8まで何を学んできたかざっと振り返ってみます。

💻week1～8で何を学んだか？

（

・week1,2：生成AIやノーコードを使った簡易アプリの開発体験

・week3～6：プログラミングの基礎

（Git、HTML, CSS, JavaScript, React, Next.js, Supabase）

・week7,8：Next.jsとSupabaseを使ったアプリの開発体験

）

week8まで、ざっくりフェーズが３つありました。

week1,2では生成ＡＩやノーコードを使って、プログラミング言語の知識が無い状態でもアプリを作成する体験をしました。

week3から6では、本格的なプログラミングを行うための基礎知識を学習しました。コードを効率的に管理するGitHubの扱い方、HTML・ＣＳＳといったフロントエンド周りの理解から始まりましたね。そしてJavaScript、Next.jsという流れで実践的なプログラミング言語を学び、データベースではSupabaseを学びました。

week7,8では、技術スタックをNext.jsとSupabaseとして、v0も活用しながらアプリ開発を体験しました。

正直、簡単では無かったと思います。しかしこれで、ついにあなたも自分でアプリ作成ができる土台が整いました。

そこでweek9から12では、実際にあなたが転職活動の際に用いることのできるポートフォリオを作成します。

まず、week9全体の流れはこちら。

💻week9全体の流れ

（

①要件定義：誰の何の課題を解決するアプリを作成するか決める（WhyとWhat）

②設計：そのアプリを作るための設計を行う（How）

③初期設定：開発を効率よく進めるための設定を行う

）

実際のシステム開発の現場では、「要件定義」と「設計」という、コーディングを始める前の工程が発生します。ここは深めると奥が深く難しい領域で、おそらく未経験エンジニアとして就職した直後は担当しないと思います。しかしポートフォリオ作成のためには避けて通れない道ですし、採用担当者に熱意を示す一要素としても有効ですので、week9はここに集中して学習・実践をします。最後の「初期設定」ですが、ざっくりいうとGitHubの設定やv0でプロトタイプ作成などを予定しています。week10以降で開発をしやすくするための前準備のようなイメージですね。

次に「ポートフォリオ」についてです。転職を目指されているならご存じかもしれませんが、念のため用語の確認をさせてください。

💻エンジニアのポートフォリオとは？

（

【一般的な意味】

・目的：自分のスキルや実績を証明することで、採用担当者に人材価値をアピールする

・内容：開発成果物、保有スキル

【未経験エンジニア転職にとって】

・そもそも未経験の場合は、単純なスキル・実績ではなく「学習意欲」・「熱意」などのポテンシャルも重要視される

・ポートフォリオが必須でない場合もある

・「同レベルの人より優位に立つ」、「学習意欲を示す」といった観点でポートフォリオは有効

）

エンジニアにおけるポートフォリオとは、開発成果物や保有スキルをまとめたものとなります。もしエンジニア経験があれば、「Pythonを3年やっていました」といった保有スキルの説明や、「基幹システムのフロントエンド開発をしていました」といった実績を書くことができます。しかし、未経験の場合にはそういったものが無いので、「自分で作ったアプリを見せる」が一番です。未経験の場合はそもそもポートフォリオが必須で無い場合もありますが、「同レベルの人より優位に立つ」「学習意欲を示す」ためにも、ぜひこの講座で一緒に作っていきましょう。

最後に、week9から12までで主に使う技術スタックを確認します。

💻本講座で想定している技術スタック

（

・フロントエンドとバックエンド：Next.js

・データベース：Supabase

・バージョン管理や資料の格納：GitHub

・デプロイ：Vercel

）

いずれも、本講座で学習してきた内容になります。Next.jsの中にはHTML, CSS, JavaScript, Reactといった要素も入ってくるので、学んだことをフルで活かして頑張っていきましょう。もしご自分で他に学ばれているものがあれば、もちろん使っていただいても構いません。

それでは、本動画のまとめに入ります。

💻まとめ

（

・week8までで、簡易アプリ作成→基礎知識理解→Next.jsとSupabaseによる実践的なアプリ作成

・week9～12では、ポートフォリオのためのアプリ作成をそれぞれで行う

・week9ではプログラミングの前準備として、「要件定義→設計→初期設定」を行う

・技術スタックは、week8まで習った内容に留める

）

※ほぼそのまま読む

それでは次回から、「何のためにどんなアプリを作るの？」という要件定義に入ります。ありがとうございました。

## ②要件定義1\_ペルソナ設計

こんにちは。これから、あなたが自分のアプリを作るための要件定義を進めていきます。

本日の流れはこちら。

💻本動画の流れ

（

・要件定義とは

・ペルソナ設計とは

・ペルソナ設計の実践

）

まず「要件定義」とは何かを理解したうえで、要件定義の第一段階である、「誰の課題を解決したいか？」を決めます。この後の課題で、自分でペルソナ設計をしてもらうので、ぜひ自分事のように受講してください。

それではまず「要件定義」とは、何か簡単にご紹介します。

💻要件定義とは

（

・目的：システム開発の目的を明確にし、開発範囲や内容を決定する

・やること：システムに求められる機能や性能、条件などを明確にする作業

・成果物：要件定義書（ドキュメント）

）

要件定義とはざっくりいうと、「ある課題を解決するために、こんなシステムを作ります」というのを決める作業です。さらにざっくりいうと、システム開発における「Why」と「What」を決める作業とも言えます。

どんなシステムも、基本的には誰かの何かの課題を解決するためのもので、それがwhyです。whatは、その課題を解決するために「何を作るのか」ということですね。「どう作るか」というhowはこの後の「設計」で行うので、今は考えなくて大丈夫です。

ちなみにこの要件定義、実は超重要です。

💻要件定義の重要性

（

【共通】

・“なにを作るか”がはっきりすることで、開発のブレがなくなる

・優先順位がつけられるので、途中で慌てずに実装できる

【受託開発の場合（例：顧客から「○○の課題を解決したい」と言われた）】

・顧客の期待値を調整し、顧客が納得できるシステムを作ることができる

・顧客の予算と開発コストのすり合わせができ、正確な見積もりを作ることができる

→これがないと、「思ってたのと違う…」と納品後に揉める可能性が高くなる

【自社開発の場合（例：とある課題を解決できるアプリを自社で0から作る）】

・誰（ペルソナ）のどんな課題を解決するかを、エンジニア以外の人も理解できる

→これがないと、「作ってみたけど売れない」と使ったリソースが無駄になる可能性が高くなる

）

まず全般的なメリットですが、「何のために何を作るか」がはっきりすることで、その後の開発がスムーズになります。

次に個別のメリットですが、「受託開発」と「自社開発」に分けて考えます。

「受託開発」は例えば、「ウチの会社のホームページを作ってよ」や「ウチの業務を効率化できる基幹システムが欲しい」など、顧客から予算を頂いて開発を行う形態ですね。この場合はペルソナや課題は概ね明確なのですが、重要なのは顧客の納得度です。例えばあなたが依頼側だとして、本当はスマホとＰＣ両方に対応できるシステムを作ってほしかったのに、ＰＣ版だけ作られたらどう思いますか？　「思ってたのと違う…」となりますよね。要件定義をしっかりすれば、そういった事態を防ぐことができます。

次に「自社開発」ですが、皆さんが良く知っているようなアプリは大体そうです。NetflixやYouTubeやLINEなど、自社で開発したものを全世界に公開して、収益を得ていますね。この場合は、開発時点で目の前に顧客がいるわけではないので、ペルソナ設計が必要になります。

ちなみに本講座では、皆様独自のアプリを作成していただくために、「自社開発」の意識で、開発を進めていきます。

それではペルソナ設計に入りましょう。

💻ペルソナとは？

（

・意味：商品やサービスを利用する架空のユーザー像

・目的：施策や開発したシステムの効果を上げるため

・活用場面：自社開発だけでなく、採用、マーケティングなど様々なビジネスシーン

）

そもそもペルソナとは？　という話ですが、エンジニアが未経験の皆さんでも、聞いたこと・考えたことがあるかもしれません。というのもペルソナの活用場面は広く、採用やマーケティングでもよく使われるからです。

この講座でそこまでやるの？　と思われるかもしれません。しかし、ポートフォリオの評価軸の一つに、「本当にそのシステムは課題を解決できるか」というものがあります。例えば二人の候補者が同じ技術レベルだとして、「ブログアプリ」と「誰かの特定の悩みを解決できる唯一のアプリ」を作っている人、どちらを取りたいですか？　後者ですよね。

ですので、皆さんの転職確率を少しでも上げるために、ペルソナ設計からやっていきます。とはいえ本気で作ろうとすると、かなり時間がかかります。そこで本講座では、最低限で進めます。

💻本講座における「ペルソナ設計」に含める項目

（

| 項目 | 例 |
| --- | --- |
| 自分との関係性 | 友人 |
| 名前/年齢/性別 | TK、32歳、男性 |
| 職業/働き方 | 公立高校の教師（社会）、男子バスケ部の顧問 |
| 日常的な行動習慣 | 平日：  ・朝7時登校／朝練指導（週3）  ・1日3～5コマ授業＋委員会や担任対応  ・放課後は部活で19時頃まで指導  ・その後、職員室で課題チェックやプリント作成で帰宅は21〜22時が多い  休日：  ・土日は大会や練習試合、顧問会議などで1日外  ・月1〜2回しか完全休養がない |
| デジタルリテラシー | スマホは慣れている  PCはOffice 365やGoogle Driveを触れる程度 |
| 悩み・課題（想定でも可） | ・社会の授業プリントや小テストの内容を改善したいが、時間が無くて毎年使いまわし  ・部活メニューを考えるのが地味に手間  ・他校のバスケ部顧問と話してもっと上手く部活を回したい |

※「今の自分」・「過去の自分」・「家族や友人」・「会社の同僚」・「会社の顧客」など、身近な人からだと考えやすい

）

左に項目、右に例を書いてみました。職業/働き方までは基本情報ですね。開発の方向性に直結するのは一番下の「悩み・課題」ですが、デジタルリテラシーも重要です。やや複雑なシステムでもいいか、なるべく操作を簡単にすべきかなどを決めやすくなります。

日常的な行動習慣は、例ほど詳しく書かなくても構いません。その人の一日をざっくりとイメージできればＯＫです。

ここまで書いたらどうでしょう？　こんなアプリがいいなぁ？　と見えてきませんか？

詳しくは次回話しますが、例えばこんなアプリが思いつきます。

💻悩み→アプリ化の例

（

| 悩み | アプリ |
| --- | --- |
| ①社会の授業プリントや小テストの内容を改善したいが、時間が無くて毎年使いまわし | 画像を添付するだけで、改善点とその理由を教えてくれるアプリ  ←生成AI必須、ログイン任意 |
| ②部活メニューを考えるのが地味に手間 | ボタンタップ（スポーツ・時間・人数など）だけで、今日の部活メニューの案を出してくれるアプリ  ←生成AI必須、ログイン任意 |
| ③他校のバスケ部顧問と話してもっと上手く部活を回したい | 部活顧問に特化した匿名SNSアプリ  ←ログイン必須、複数ページ必須 |

）

例えば授業資料や小テストの改善は、生成AIですね。皆さんも使って実感されたと思いますが、ChatGPTなどの生成AIを使いこなすには、上手くプロンプトを入れる必要があります。これがけっこう難しいので、ユーザーが画像添付だけで良い指摘が返ってくるアプリはアリかと思います。

それでは、本日のまとめです。

💻まとめ

（

・要件定義とは「何を作るか」・「なぜ作るか」を決めること

・要件定義をしないと、「開発中の後戻り」や「顧客の期待値とのズレ」が発生しうる

・ペルソナとは「商品やサービスを利用する架空のユーザー像」

・ペルソナを決めることで、具体的なアプリ像が思い浮かぶ

）

※ほぼそのまま読む。

ちなみに講義後の課題では、実際にペルソナ設計をしていただきます。テンプレートつきのスプレッドシートをお渡しするので、最低１人分作成してください。悩んだら、ChatGPTに「この項目でペルソナ設計をして」などと頼んでみてもいいでしょう。

それではお疲れ様でした。次回は作ったペルソナを元に、「どんなアプリがいいかなぁ」と要件定義を進めていきます。ありがとうございました。

## ③要件定義2\_ソリューション検討（要求定義）

こんにちは。今回は、前回作ったペルソナに対して、どんなシステムで解決できるかを考えます。

ここがブレると、今後の設計も実装も全てが的を得ないものになってしまいます。転職時の面接でも、「なぜこのアプリを作ったの？」という質問が来る可能性があり、ここでしっかりと悩みとアプリを論理的に結びつけることで、上手く答えられるようになります。

それでは本日の流れはこちら。

💻本動画の流れ

（

・【解説】本講座で取り扱うソリューションの範囲

・【実践】ペルソナの悩みからソリューションを考える

・【実践】類似サービスの検索、ブラッシュアップ

）

まず、本講座で取り扱うソリューションの範囲を簡単に解説します。例えば、YouTube並の大きなアプリは難易度が高く、エンジニア未経験のポートフォリオとしてもそこまで求められません。

その後、実際にソリューションを検討していきます。それではまず、ソリューションの範囲から。

そもそもなのですが、システム開発ってどんなものがあるのか？　から解説します。

💻システムの大分類

（

本講座では、一つ目の「Webアプリ」のみを扱います

| 種類 | 概要 | 例 | 難易度 |
| --- | --- | --- | --- |
| Webアプリ | ブラウザ上で動作。ChromeやEdgeの中で使うアプリ | week7,8で作ったアプリ、会社のHP | ★～ |
| 連携型アプリ | 既存のアプリと連携するアプリ | LINEチャットボット | ★～ |
| モバイルアプリ | スマホにインストールして使うアプリ | LINE、PayPay、スマホゲーム | ★★～ |
| デスクトップアプリ | PCにインストールして使うアプリ | Word、VS Code | ★★～ |
| 拡張機能型アプリ | ブラウザなどの機能を拡張するアプリ | Chrome拡張機能（翻訳、広告ブロック） | ★★～ |
| CLI・スクリプト | コマンドで動く | ローカルで動く自動処理ツール | ★★～ |
| API・ミドルウェア・ライブラリ | 他システムの中間的存在 | Reactそのもの、データ処理ができるNumpy | ★★～ |
| 組み込み系/IOT | ハードウェアを動かすプログラム | 家電 | ★★★★～ |

）

かなり大きな表になりましたが、大分類でもこの種類の多さになります。一番右の難易度は、最も簡単なものを作るとしたら、を基準にざっくりと設定しています。

本講座では、いろいろあるシステムの中でも、Webアプリに限定して説明します。難易度が低いのはもちろん、アプリが目に見えるのでポートフォリオとして優秀です。もちろん、やる気がある方は独学でスマホアプリなどを作っていただいても構いません。

ちなみに、もう一つの難易度１の連携型アプリは、LINEチャットボットや、ノーコードでできるようなことです。こちらも簡単なのですが、ポートフォリオとしてはやや見劣りするので講座では取り扱いません。

次に、もう一つ条件を増やします。

💻本講座で取り扱う技術スタック

（

・フロントエンド：Next.js（React）

・バックエンド：Next.jsまたは**Python**

・データベース：PostgreSQL（Supabase）

・インフラ（デプロイ）：Vercel

・バージョン管理：Git, GitHub

・開発環境：VS Code

※技術スタック：アプリを開発するために使用される技術やツールの組み合わせ

）

それは、技術スタックです。基本的にはここまでの講義で扱った内容で、逆にこれ以外は扱いません。

よく見ると、バックエンドにPythonを増やしていますね。これは、特定の機能をつけたいときに、Next.jsだけでは完結できない可能性があるためです。Pythonなら、画像処理など様々な機能をライブラリから簡単に実装できるため、Next.jsの補完として少し扱います。

ではこれを念頭に、ソリューションを検討してみましょう。まずはソリューション検討のためのTipsを確認します。

💻ソリューション検討のTips（ポートフォリオのために）

（

・一つの課題に絞る

・「既存ツールで代替できないか？」と独自性を検討する

・「技術スタック的に実装可能か？」と実現可能性を検討する

）

コツは三点です。解決する課題を絞り込み、本当に作る価値があるかを調べ、自分に作れるかを確認します。

これを念頭に、前回例で出した教師のペルソナからソリューションを検討してみましょう。

💻ペルソナ→ソリューション検討

（

| 悩み・課題 | ソリューション案 | 既存ツールで代替できないか？ | 技術スタック的に実装可能か？ | 総評 |
| --- | --- | --- | --- | --- |
| 授業資料改善 | 画像添付だけで改善点がでる | ChatGPTで代替できるが、質で勝負できる | 可能。ただし全教科対応ならプロンプトが難しい | ★★★  独自価値、実装可能性ともに高水準 |
| 部活メニュー | ボタンタップだけでメニューを作れる | ChatGPTで代替しやすい | 可能。比較的簡単 | ★  簡単だが代替可能性が高い |
| 顧問と繋がりたい | 部活顧問特化の匿名SNS | 専用SNSは存在しない | 可能。ただし複数ページ必須なのでやや難しい | ★★  独自価値はあるかもしれないが実装につまづくリスクあり |

）

悩みとソリューション案に対して、検討をしてみました。

例えば、授業資料改善。ChatGPTでも代替できそうですが、AIへの指示を工夫すれば質を上げられそうです。ちなみに、実際に類似サービスがないか調べてみましたが、ありませんでした。実装可能性も、OpenAIのAPIなどを使えばできそうですし、１ページだけのアプリでも済みそうです。

他の二つは、独自価値が低かったり、実装が難しそうなので総合評価を下げています。

このようにして、「じゃあ授業資料改善アプリを作ろう」と決めることができます。

ちなみに、各項目の検討にはChatGPTの利用を推奨します。悩みからソリューション案を出させるもよし、類似サービスが無いか検索するもよし、表を作り切った後にフィードバックを貰うも良しです。

それでは本日のまとめに入ります。

💻まとめ

（

・社会には様々な種類のシステムがあるが、本講座では「簡単さ」と「ポートフォリオとしての価値」からWebアプリのみを取り扱う

・技術スタックは、「これまで講義で扱ったもの」＋「任意でPython」

・ソリューションの絞り込みは、「課題の解決性」「独自性」「実装可能性」で行う

・生成AIを積極的に活用して、よりよいソリューションを考える

）

※ほぼそのまま読む。

ちなみに講義後の課題では、実際にソリューション検討をしてもらいます。最終的に、あなたがポートフォリオに載せるためのアプリを決められればOKです。

それではお疲れ様でした。次回は、絞り込んだソリューションについて、「どんな機能がいるか」などを一緒に考えていきます。ありがとうございました。

## ④要件定義3\_機能要件・非機能要件を整理する

こんにちは。今回は、前回絞り込んだソリューションを、もっと具体的にしていきます。本日の流れはこちら。

💻本動画の流れ

（

・「機能要件」「非機能要件」とは？

・「機能要件」と「非機能要件」の重要性

・ジャンル別のテンプレの紹介

）

「機能要件」と「非機能要件」というのが、具体化する方法です。新しい単語なので、例を含めて分かりやすく解説します。その後、皆さんが実践しやすくするために、ジャンル別の例もご紹介します。

まずはイメージと例を解説します。

💻「機能要件」・「非機能要件」とは

（

|  | 機能要件 | 非機能要件 |
| --- | --- | --- |
| 一言で | 何ができるか？ | どうあるべきか？ |
| 簡単な定義 | アプリに必ず必要な機能 | 機能要件以外に、開発で考慮すべき要件 |
| 飲食店のオーダーアプリの例 | ・メニューを表示する  ・料理を選んで注文する  ・注文履歴を確認できる | ・料理を美味しく見せる  ・注文完了まで3ステップ以内  ・スマホ画面に最適化 |

※機能要件→非機能要件　の順に考える

）

機能要件とは、アプリに必ず必要な機能です。例えば、注文ボタンがなければ飲食店のオーダーアプリとして明らかに不完全ですよね。

一方で非機能要件は、機能要件以外に考慮すべき要件です。例えば「料理をおいしく見せたい」、「PCじゃなくてスマホ画面に最適化」など、必須ではないがユーザーの満足度を上げるために考慮したいこと全般を指します。

では、なぜこの工程が必要かも、少し触れておきます。

💻「機能要件」・「非機能要件」の重要性

（

・「要件Aを入れるか入れないか」で、実装難易度が大きく変わる

・設計や開発時の「手戻り」を防ぐ

）

まず実装難易度についてです。例えば、「注文履歴を確認できる」という機能要件を「入れない」と決めた場合、そのページや裏側のプログラムが不要になりますよね。他にも、「多言語対応にする」という機能要件を追加するなら、必要なコードが増えます。

このように、機能要件・非機能要件を決めることによって、実装の難易度を概ね確定させることができます。これができれば、二つ目に書いてある通り、手戻りも防げます。

講義後の課題で実際に考えてもらうのですが、オーダーアプリの例では不十分かと思います。そこで、ジャンル別の例をご紹介します。

💻ジャンル別の要件例

（

| ジャンル | アプリ例 | 機能要件例 | 非機能要件例 |
| --- | --- | --- | --- |
| 情報まとめ系 | 日経ビジネス電子版 | キーワード検索、ブックマーク | 見やすい、画像つき |
| ソーシャル系 | X、Instagram | 投稿、グッドボタン、フォロー機能 | 様々なアカウントでログイン可能 |
| 特定課題解決系 | 文字数カウント | 課題による | なるべく短いアクションで使える |
| 業務支援・管理系 | Todoリスト | データ入力、履歴確認 | グラフは色付きで見やすくする |

）

おそらく皆さんが作るアプリは、表に書いたジャンルのどれかだと思います。

ジャンル一つ目は、情報まとめサイトですね。温泉でも猫の品種でも、様々な切り口で情報をまとめたサイトです。このジャンルは、「検索」のしやすさがカギになります。例えば日経ビジネス電子版では、キーワード検索だけでなく、カテゴリ別の検索もできます。しかも、各記事が何分で読めるか、という非機能要件も入っています。

二つ目はソーシャル系ですね。前回でいうと部活顧問SNSアプリ、他にはweek8で作ったブログアプリもこのジャンルに含まれます。皆さんも例のうちどれかは使っていると思いますが、「投稿できる」「他人の投稿に何かしら反応ができる」ことはほぼ確実に必要ですよね。それが機能要件に入ってきます。例えばＸなら、グッド以外にリポスト、ブックマークもできます。

三つ目は特定課題解決系です。これは幅が広いので機能要件は場合によりけりです。前回でいうと授業資料改善アプリや、部活メニュー作成アプリはここに含まれます。独自性を強く出せるので、ポートフォリオ向きだと思います。ちなみに非機能要件は概ね共通していて、「簡単にすぐ使える」ことが重要です。例えば文字数カウントは、サイトに入って文章を打つだけですぐ結果が出てきます。

最後のジャンルは、業務支援・管理系ですね。week7のtodoリストはこのジャンルに入ります。他でいうと健康記録やカレンダーもこのジャンルです。このジャンルは、最低でもデータ入力と一覧表示は機能要件に入ります。非機能要件は場合によりけりですね。例えばユーザーを「40代のサラリーマン男性」に絞るなら、色味はピンクではなく青のほうがいいでしょう。

それでは、本日のまとめに入ります。

💻まとめ

（

・機能要件＝アプリに必ず必要な機能　（≒何ができるか？）

・非機能要件＝機能要件以外で考慮したい要件　（≒どうあるべきか？）

・この工程で、実装難易度がだいたい決まる。手戻りを防ぐためにも重要

・実践では、ジャンル別の例を参考にしながら作成する

）

※ほぼそのままよむ。

ちなみに講義後の課題で、実際に機能要件・非機能要件を作成してもらいます。細かければ細かいほど次が楽になりますが、箇条書きでそれぞれ３行程度でも構いません。

それではお疲れ様でした。これで「何を作るか」「なぜ作るか？」の要件定義は完了です。次回以降は「設計」つまり「どう作るか」という工程に入ります。まだコードは書きませんが、質の高いポートフォリオのために重要な工程なので、一緒に頑張っていきましょう。ありがとうございました。

# アプリを具体的に「設計」する

✅基本設計、詳細設計の要点・流れを理解し、実践

## ①基本設計1\_基本設計とは？

こんにちは。ここからは、「設計」という工程に入ります。前回までで、「誰のどんな課題を、概ねどんなアプリで解決するか」を決めました。もうプログラミングに入れるのでは？　と思ったかもしれませんが、実務ではもう一段階工程があります。それが「アプリをどう作るか？」という設計工程です。それでは本日の流れはこちら。

💻本動画の流れ

（

・システム開発の全体像をおさらい

・基本設計とは？

・基本設計の重要性

）

まず、改めてweek9から12で体験するシステム開発全体のフローをおさらいします。その後、設計の第一段階、基本設計について理解を深めます。

それでは、全体フローのおさらいから始めましょう。

💻システム開発の全体像　※フロー形式。一旦表で書くけど

（

| 要件定義 |  |  | 設計 |  | 開発 | テスト・デプロイ | 改善 |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ペルソナ設計 | ソリューション検討 | 機能要件・詳細要件 | 基本設計 | 詳細設計 |  |  |  |
| 誰のどんな課題を解決するか？ | その課題をどう解決するか？ | 必要な機能、その他要件は？ | 非エンジニアも見れるアプリの設計書 | エンジニアが開発のために使うアプリの設計書 | コードを書いてアプリを作成する | コードを修正し、公開する | アプリを改善する |
| week9 |  |  |  |  | week10 | week11 | week12 |

）

まず、一番上にあるように、要件定義→設計→開発→テストとデプロイ→改善が大まかなフローです。実際コードを書き始めるのは開発からですね。

week9では、このうち要件定義と設計を取り扱います。それ以降は、フローごとにweekを分けて取り扱います。そして、既に要件定義は完了しています。ここからは「どうつくるか」、つまり設計を行います。

設計は最終的に、「よしこれでプログラミングに取り掛かれる」とあなたが思える状態になることが目的です。「あれ、テーブルのカラムどうするんだっけ」「このバックエンド処理ってNext.jsでできるんだっけ」みたいな疑問をこの工程で潰しきりましょう。

それでは次に、基本設計について理解を深めましょう。特に、詳細設計との違いが重要なので、比較しながら解説しますね。

💻基本設計とは

（

| 項目 | 基本設計 | 詳細設計 |
| --- | --- | --- |
| イメージ | 「何をつくるか、どんな構成で動くか」を整理する | 「どうやって作るか、どのように処理するか」を詰める |
| 目的 | 非エンジニアと合意できる（社内、社外問わず） | エンジニアが迷いなく開発に取り掛かることができる |
| フロントエンド | 画面の種類を整理 | 各画面のデザインと遷移構造まで決める |
| バックエンド | 機能ごとの役割や繋がりを整理 | 使用するライブラリ、データ引き渡しの型まで決める |
| データベース | どんなテーブルがあるか | テーブルのカラムやリレーションまで決める |

）

まず基本設計ですが、これは非エンジニアにも見せられるものです。例えば受託開発ならクライアントにも見せて「本当にこういうアプリでいいですか？」と最終確認を取ることになります。「こんな画面の種類があって」「こんなことができて」「こんなテーブルがある」くらい決めればOKです。

逆に詳細設計は、エンジニアのためのものです。例えばフロントエンドなら「ログインボタンは何色にするか？」くらいの細かさまで決めます。バックエンドなら、使うライブラリや、ここはjsonデータでやり取りをする、くらいまで決めます。データベースでも、例えば「あれ、カラム名なんだっけ、カラムのデータ型なんだっけ」という悩みは全て潰します。

さて、内容は分かったけど、本当に必要なの？　すぐコーディングに入った方が速いんじゃないの？　と思われるかもしれません。そこで次に、基本設計の重要性を確認します。

💻基本設計の重要性

（

・後戻りが減る（設計ミスや漏れに早く気づける）

・非エンジニアと深い合意を作ることができ、開発後に不満が発生しにくくなる

・チームで意思疎通が取りやすい（共有しやすい）

・面接で「設計の意図」まで話せる（レベルが高く見える）

）

概ねこの４点です。

以前から「後戻りが減る」を何回も言っていますが、システム開発は工程が多く、本当に「後戻りが減る」というのは重要なんです。

加えて重要な観点が、「コミュニケーションツールとしての基本設計書」です。非エンジニアとの合意はもちろん、開発チームでも話がしやすくなります。他にも、転職時の面接で意図を聞かれたときに話しやすいですね。

それでは、本日のまとめに入ります。

💻まとめ

（

・システム開発全体のフローは「要件定義」→「設計」→「開発」→「テストとデプロイ」→「改善」

・設計は「基本設計」と「詳細設計」に分かれる

・基本設計は、非エンジニアも見て合意することができるもの

・詳細設計は、エンジニアが迷いなく開発ができるようになるもの

）

※ほぼそのまま読む

お疲れ様でした。次回は基本設計の実例をお見せします。それでは、ありがとうございました。

## ②基本設計2\_基本設計の実践

こんにちは。この動画では、実際に皆さんが作る基本設計書の中身をみていきます。本日の流れはこちら。

💻本動画の流れ

（

・基本設計書に書くべき内容（成果物）をパート別に整理

・パート別、ジャンル別に具体例を紹介

）

成果物の整理と具体例の紹介、という二段構成でいきます。

それではまず、作るべき成果物を整理します。

💻基本設計で作るべき成果物

（

| 項目 | 作るべき成果物 | 簡単な例 |
| --- | --- | --- |
| フロントエンド | 必要な画面の種類、画面間のつながり | ・画面一覧：トップ、新規登録、ログイン  ・繋がり：トップ画面→ログイン画面に遷移できる |
| バックエンド | 実装予定の機能一覧、機能の目的 | ・機能一覧：ユーザー登録、投稿作成 |
| データベース | 必要な情報の種類（テーブル名と概要レベル） | ・Userテーブル、Postテーブルなど |
| その他 | 技術スタックなど | Next.js、Supabaseなど |

※全て「テキスト」形式

）

成果物と簡単な例を、パート別に表にしてみました。

フロントエンドでは、どんな画面があるか、画面同士はどうつながるかを書きます。バックエンドでは、どんな機能があるかを漏れなく整理します。データベースは、テーブル名程度で良いです。その他は、技術スタックなど、詳細設計を考える際に必要な情報を整理しておきます。

ちなみに基本設計は、全てテキスト形式で大丈夫です。詳細設計では、画面の絵やテーブルの図解も必要になりますが、基本設計ではそこまでやりません。

次に、実例をご紹介します。

💻パート別×ジャンル別の基本設計例

（

| パート\ジャンル | 情報まとめ系 | ソーシャル系 | 特定課題解決系 | 業務支援・管理系 |
| --- | --- | --- | --- | --- |
| フロント | ・トップ画面（検索バーあり。記事がカードとして一覧表示）  ・記事詳細ページ  ・カードタップで記事詳細ページへ遷移 | ・トップ画面  ・投稿作成画面  ・マイページ画面  ・トップ画面から、全画面に遷移できる | ・トップ画面のみ | ・トップ画面（データ一覧）  ・データ作成画面  ・トップ画面からデータをポップアップで編集・削除できる |
| バック | ・記事のブックマーク | ・投稿作成  ・ユーザー名変更  ・Google ログイン可能 | ・テキスト解析（文字数カウント）  ・生成AI連携（部活メニュー） | ・データのCRUD全て |
| DB | ・users, articles  ・ブックマークはusersテーブルに含める | ・users, posts  ・comments, categoriesなどは作らない | ・post（ユーザーが入力した内容と出力結果） | ・users, tasks  ・一対多リレーション |
| その他 | ・キーワード検索について実現方法を調べる必要あり | ・X(旧Twitter)を参考にする | ・なるべく早く結果が返せる技術スタックに | ・バックエンドもNext.jsのみで完結予定 |

）

情報量がかなり多いですが、こちらが実例です。正直、これでも情報量が少ないです。最低でも、各パート３行以上は欲しいところです。

さて、フロントエンドはご覧の通り、どんな画面があるか、そして画面はどうつながるかですね。バックエンドは、ユーザー目線で書けると良いです。データベースは、ここではテーブル名程度でいいでしょう。もし検討したけど作らないテーブルがあれば、理由と共に記載しましょう。

その他は、場合に応じていろいろ書けます。例えば必須機能だけど実現できるか分からないなら、申し送りとして「詳しく調べる」などと書いておきましょう。他に、画面や機能で真似できるアプリがあればイメージがつきやすくなります。非機能要件や技術スタックも特記事項があれば書いておきましょう。

それでは、本日のまとめです。

💻まとめ

（

・基本設計で作るべき成果物：テキスト形式でパート別に整理

・成果物の具体例：パートごとに三行以上は書く

・その他に含めるべき内容：申し送り、参考アプリ、技術スタックの特記事項など

）

※ほぼそのまま読む

お疲れ様でした。今回の課題では、実際に基本設計書を作成してもらいます。そしてそれを元に、次回から詳細設計を進めます。それでは、ありがとうございました。

## ③詳細設計1\_詳細設計とは？

こんにちは！

この動画では、「詳細設計ってなに？なんでやるの？」を明らかにしつつ、実際の成果物を一緒に見ていきます。本日の流れはこちら。

💻本動画の流れ

（

・「詳細設計」とは\_実装前の最終準備

・詳細設計で作る成果物の例

）

詳細設計はシンプルにいうと、実装フェーズの前にやる最終準備です。なぜ必要か、何を作れば最終準備として完了かをこの動画では解説します。

まず詳細設計とは何か、基本設計のときに見たスライドを振り返りながら解説します。

💻詳細設計とは\_実装前の最終準備

（

| 項目 | 基本設計 | 詳細設計 |
| --- | --- | --- |
| イメージ | 「何をつくるか、どんな構成で動くか」を整理する | 「どうやって作るか、どのように処理するか」を詰める |
| 目的 | 非エンジニアと合意できる（社内、社外問わず） | エンジニアが迷いなく開発に取り掛かることができる |
| フロントエンド | 画面の種類を整理 | 各画面のデザインと遷移構造まで決める |
| バックエンド | 機能ごとの役割や繋がりを整理 | 使用するライブラリ、データ引き渡しの型まで決める |
| データベース | どんなテーブルがあるか | テーブルのカラムやリレーションまで決める |

※ただし詳細設計に必ず従う必要は無く、開発の中で変更が生じれば詳細設計を書き換えてよい

）

まず、基本設計では、どんな画面やテーブルを用意するか、どんな機能があってどうつながっているかをテキスト形式で洗い出しましたね。

しかし、これだけで開発にすぐ取り組めるでしょうか？　例えば「ログインボタンはどこに配置する？」とか「user idの形式は何にする？」とか、まだ迷いが生じてしまいます。詳細設計では、そのような疑問を潰します。

ただし、「詳細設計を絶対に守る」ことは必須ではありません。開発を進めていくと、この機能が実は難しすぎる、画面遷移は違う方がいい、などの気づきが発生するでしょう。そうなったときは、詳細設計書を書き換えてOKです。

それでは、成果物の例をパート別に見てみます。

まずはフロントエンド。画面の構成や遷移を表した、ワイヤーフレームというものを作ります。

🗺️（「figma ワイヤーフレーム　例」で検索し、何個かピックアップしながらしゃべる）

実際の例はこちら。figmaというアプリを使って作成されたワイヤーフレームです。全ての画面、要素の色や配置、画面遷移など、全ての情報が書かれています。ここまで書けると、実装しやすいですよね。

とはいえ、figmaは学習コストが高いので、今回は手書きでも大丈夫です。手書きの例はこちら。

🗺️（「ワイヤーフレーム 手書き」で検索し、なるべく簡単そうなのを何個かピックアップしながらしゃべる）

手書きでも、要素をどう配置するか、画面はどう遷移するかなどは十分に書けます。さらに、手書きの画像をChatGPTやClaudeに読ませることで、もっと精度の高いワイヤーフレームを作ることも可能です。

次にデータベース。テーブルの詳細とリレーションを表した、ER図というものを作ります。

🗺️（「ER図　例」で検索し、なるべく簡単そうなのを何個かピックアップしながらしゃべる）

実際の例はこちら。テーブル名、カラム名はもちろん、リレーションは線で表します。各カラムのデータ型まで指定できていると理想ですね。

こちらもツールを使って書けるのですが、手書きでも大丈夫です。

最後にバックエンド。ここは正直、分量が多いです。

🗺️（「draw.io　ワークフロー　例」で検索し、なるべく簡単そうなのを何個かピックアップしながらしゃべる）

一つ目はワークフロー図。いわゆるフローチャートですね。このdraw.ioというツールを使えば、先ほど紹介したER図もフローチャートも作れるので便利です。

ワークフローは必須では無いのですが、条件分岐が複雑で長い処理の場合には、あった方が便利です。逆に、サインアップやログインなど基本的なバックエンドには今回は不要です。

そして、バックエンドで作るべき成果物がもう一つあります。それは機能一覧です。

💻機能一覧の例

（

| ファイル名 | 目的 | 補足 |
| --- | --- | --- |
| api/signup | ユーザーがサインアップをする | Googleアカウントでのサインアップ可能 |
| api/post | ユーザーが投稿を作成する | Supabaseに、ユーザーIDと連携して投稿を格納する |
| api/gpt | OpenAIのGPTにアクセスして、結果を受け取る | 使うモデルはGPT-4o。 |

）

このように、ファイル・目的・補足事項を漏れなく書いていきます。思ったことはとりあえず補足に書く、くらいのイメージでいいです。

どうでしょうか。ここまで決められると、もうプログラミングにとりかかれそうですよね。しかも、詳細設計をしっかりと詰めることで、v0に作らせるプログラムの雛形の精度も上がります。

さて、それでは本日のまとめに入ります。

💻まとめ

（

・詳細設計は、「実装前の最終準備」

・フロントは「ワイヤーフレーム」、ＤＢは「ＥＲ図」、バックは「ワークフロー図と機能一覧」を作成する

・より細かく詰めることで、ＡＩに作ってもらう雛形の精度も上がる

）

※ほぼそのまま読む。

お疲れ様でした。次回から、フロントエンド、データベース、バックエンドそれぞれの詳細設計を実践していきます。それでは、ありがとうございました。

## ④詳細設計2\_フロントエンド

## ⑤詳細設計3\_データベース（ER図）

## ⑥詳細設計4\_バックエンド

# アプリ開発の「環境」を整える

## ①GitHubのリポジトリ作成＆READMEにこれまでの情報を入れる

## ②v0でプロトタイプ完成

## ③week9まとめ

# ※参考

要件定義～基本設計～詳細設計までの流れ

<https://zenn.dev/nyanchu/articles/27a3f95d98df45>

<https://products.sint.co.jp/ober/blog/basic-design-detailed-design>