

Blueprint Games (Spring 2025)

Your Task

Today, you will assume the role of a Blueprint project team member. You and your team will work together to define **project goals, deliverables, and features** for a nonprofit, assuming you have ten weeks to implement your plan.

Don't worry — **you won't be writing any code today**. Instead, your team will focus on understanding and discussing product needs, outlining key features, planning high-level user flows, and modeling your initial database needs.

If you don't understand every part of the prompt, that's okay. Ask your team members (and E-Board) for help! This exercise is as much about your ability to *collaborate* as it is about your ability to work towards completing a technical task.

Drumroll... Introducing the nonprofit you'll be working with today...

Penny Juice of America

Penny Juice of America is a family-owned, non-profit business founded in 2001, with over 40 years of experience in the beverage industry. They offer 100% blended fruit juice concentrates specifically designed for childcare centers, meeting FDA nutritional requirements. The concentrates are easy to prepare, cost-effective, and available in a variety of flavors, with free nationwide delivery and no minimum order beyond one case.

Despite their strong product line, they are facing a significant challenge in efficiently managing order fulfillment and delivery logistics, especially as demand grows nationwide. The current process involves handling individual orders manually, which has led to delays and inefficiencies in ensuring timely deliveries and meeting customer needs, impacting their service quality and

customer satisfaction.

To address this, Penny Juice is considering a new model:

1. Childcare centers can place orders through an online portal.
2. The administrative team receives these orders and assigns them to local distributors or delivery partners.
3. Delivery partners indicate their availability for a given order, and the team sends them order details.
4. Once the order is dispatched, the team marks it as "In Progress" on the portal, and customers are notified.



The Goal

□ Penny Juice's challenge is to create a more efficient platform that streamlines the process of connecting childcare centers with deliveries of juice concentrates. Consider the following questions:

- How can we leverage technology to simplify the ordering and delivery process, eliminating unnecessary steps?
- How can we improve the flow of information to ensure faster and more accurate order fulfillment?

The Users

- Childcare Centers (ordering juice concentrates)
- Delivery Partners (handling and delivering the juice concentrates)
- Administrative Team (managing orders and tracking deliveries)

Upon successful delivery, the team marks the order as "Complete," ensuring tracking and communication for both customers and staff.

This new approach aims to streamline order processing, reduce delays, and improve customer satisfaction across the country.

Getting started

When Blueprint partners with a nonprofit, they describe the challenge they're facing, and we're free to decide how to tackle it. Today, you're in Blueprint's place. How you choose to approach this prompt is entirely up to you and your team. Just make sure you can provide solid justifications for your decisions. There's no right answer—everything has tradeoffs. **For this project you can assume that you have 14 weeks (1 semester to complete it).**

Here are some questions to get you started.

- What kind of system do you think would fit best? Is it an app, a website? Or is it something else entirely?
- Consider the profile of a typical user. Are there any considerations that should be made when designing a new system for said user? How might these considerations affect the final deliverable?
- What features would be most valuable to the nonprofit and its end-users?
- How can you make sure it's feasible for your team to complete within the ten weeks?
- What should you focus on? What might you have to sacrifice?

When you're ready to get started, duplicate this [Google Doc](#) and share it with your team members. The doc contains a few tips on how to take notes.

Deliverables

☐ You **will not** be asked to present your final deliverables. We are more interested in seeing *how* you go about understanding and breaking down the problem than any polished final result.

Document any thoughts, drawings, or features as you work! **At the end, you will submit everything your team has come up with.**

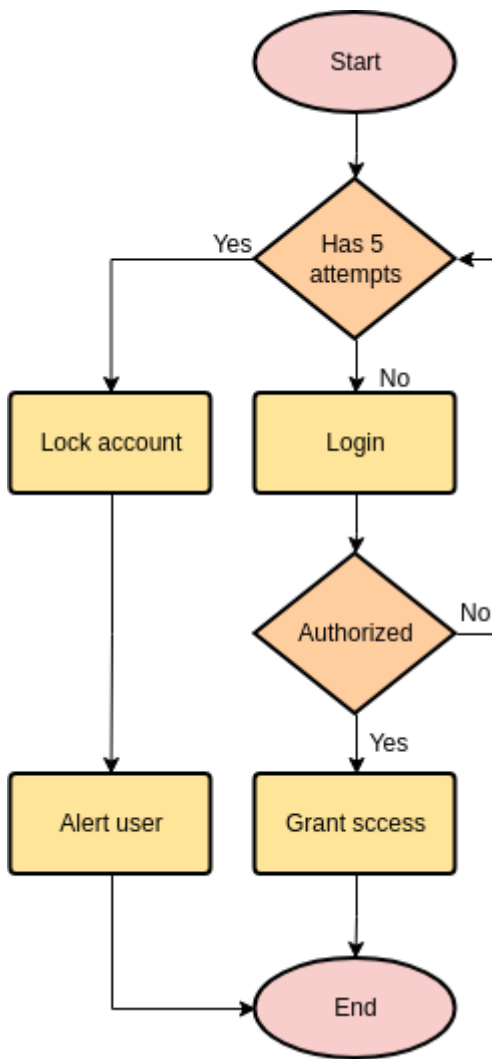
That said, we aren't looking to see a beautiful polished design doc — **we're most interested in your process**, so feel free to include in-progress work and notes, and don't fret about the formatting.

Your team will submit the design doc (created earlier) which can include any artifacts you create or use throughout this process. This can include:

- User Stories
 - A List of Key Features
 - User Flow Diagrams - include screenshots if you make them in a separate software
 - Wireframes - include screenshots if you make them in a separate software
 - Database Schema - this should include the data you think is necessary for the system to run
 - Application Programming Interface (API) Endpoints
-

Appendix

User Flow Diagram: A visual diagram that describes the steps a software system takes when a user performs performs certain actions. There can be multiple user flows, which might call for multiple diagrams or variations of diagrams. For example, say your website requires users to create an account and login. A user flow diagram for the login process would look something like this:



Each symbol means something different in a user flow diagram.

1. Ovals represent the start and end of a user flow
2. Rectangles represent actions that a user can take or actions that the system will perform
3. Rectangles represent decision points within the software

Application Programming Interface (API):

Restful API manage the state of your application. Let's say in your application you are dealing with a set of members. You need a way to bridge your frontend application with your server. We do this through API endpoints. Each endpoint defines the actions to be executed on a given resource. For instance, if you wanted to get a list of all your members the following endpoint has to be defined:

```
GET /api/v1/members
```

Now if you wanted to get a member by its ID, you'd create the following endpoint

```
GET /api/v1/members/{id}
```

To create a member you need to use the POST action and in the body of the request pass the attributes of the member

```
POST /api/v1/members
```

body

```
{
  id: "190ecfba-c6af-4639-af92-f94ad16ecb4a"
  name: "John Doe"
  team: "Marketing"
}
```

To update a member you can use the PUT action

```
PUT /api/v1/members/{id}
```

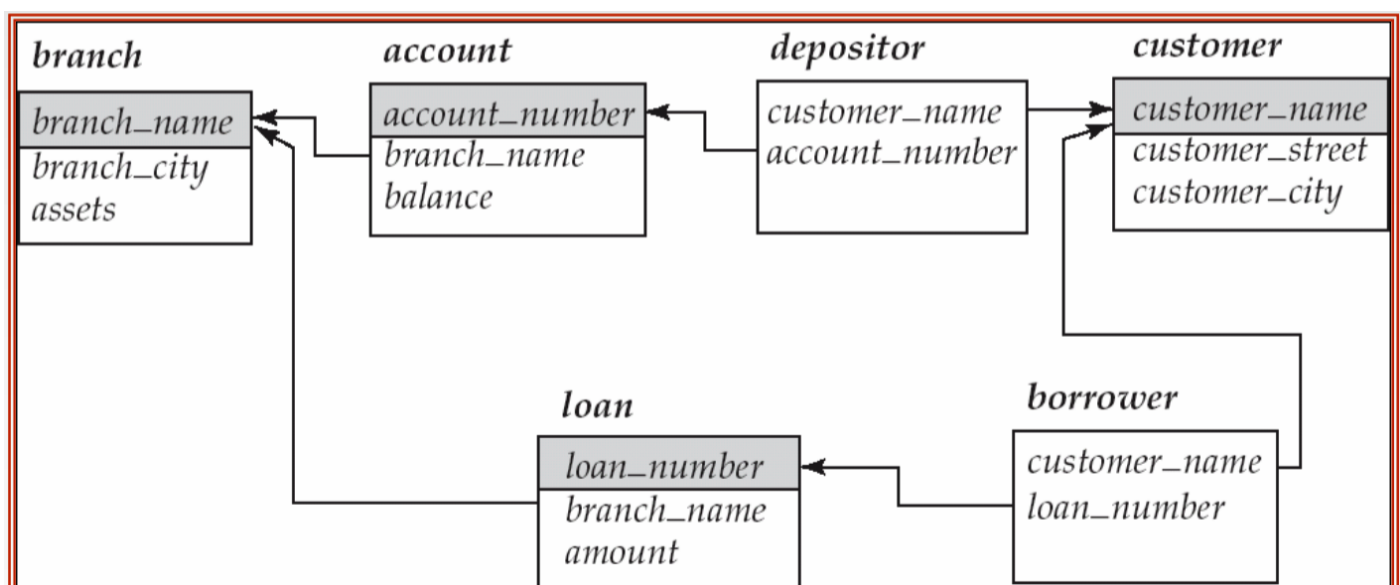
The body of the request will be a JSON with all the attributes you want to change.

Now, you can be more explicit with the behavior of the response of the endpoints. For instance, if you wanted to retrieve the members that belong to certain teamId, you can do the following

```
GET /api/v1/members/teams/{teamId}
```

In a nutshell, endpoints and its corresponding action (GET, POST, PUT, DELETE) represent a change in the state of a resource, such as a team, members, etc.

Database Schema:



SQL databases let us establish relationships between entities. For instance, a member might be related to a Team, and Team might be related to a list of members. In total, we can define 4 types of relationships: *OneToOne*, *OneToMany*, *ManyToOne*, *ManyToMany*. The way we declare this relationships is through foreign_keys.

We are looking for a simple Database Schema that defines the relationships between the entities, we do not expect you to write SQL commands or such. For instance, the db schema of the diagram shown above would be the following

```
1. branch
branch_name (PK)
branch_city
assets

2. account
account_number (PK)
branch_name (FK references branch(branch_name))
balance

3. customer
customer_name (PK)
customer_street
customer_city

4. loan
loan_number (PK)
branch_name (FK references branch(branch_name))
amount

5. depositor
customer_name (PK, FK references customer(customer_name))
account_number (PK, FK references account(account_number))

6. borrower
customer_name (PK, FK references customer(customer_name))
loan_number (PK, FK references loan(loan_number))
```

Tips

Helpful process to go about designing apps

1. What features are most important?
2. What information/data you need to build those features?
3. How should this information/data be represented, and how much of that information is shown to users?
4. What is the flow each user takes in order to access and manipulate this information?
5. What would the screens look like to each of the users?

Understanding the end user

- How can we build in a way that creates an app that is accessible, simple to use, and easy to understand?

Good luck! ☐☐

Revision #14

Created 24 February 2025 23:31:47 by Sahana

Updated 26 February 2025 20:20:42 by Lucas Ha