Software Engineering

WEEK 03 LECTURE 01

REVIEW

- Interaction with client
  - Listen your Client
  - Understand your Client
  - Involve the Client
  - Inform him about the product status
  - Take the feedback at your earliest.

SUMMARY: KEY ASPECT OF CLIENT INTERACTION

- Involve the client through meetings and continuous re-visitation of requirements.
- Use other information sources, such as end users.
- Ask good questions.

- Be assertive and open in client interactions.
- Clearly communicate realistic requirements and timelines to manage client expectations.
- Use common terms via glossaries.

- Keep track of requirements
- Remember that it is the responsibility of the software product manager to clearly communicate the pros and cons of requirements, but the client is the one who must make the key-level requirement decisions.

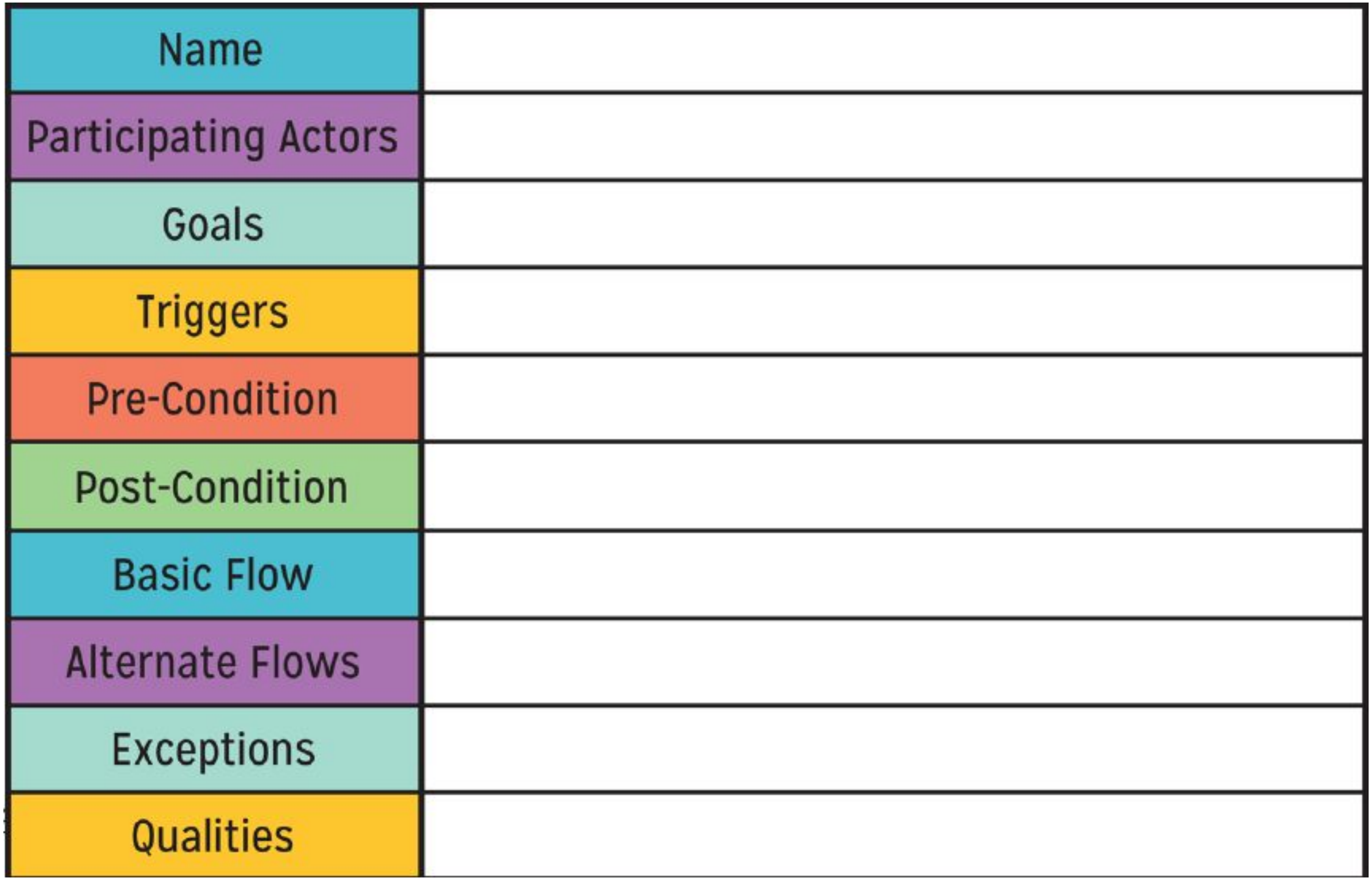
- Interviews with end-users
- Feasibility studies with focus groups
- Observing how end-users use the product
- Consulting previous products

- **Use cases**, developed by Ivar Jacobson in 1986, are a good tool for understanding a product. They can be defined as a way to identify, clarify, and organize details of a task within a product

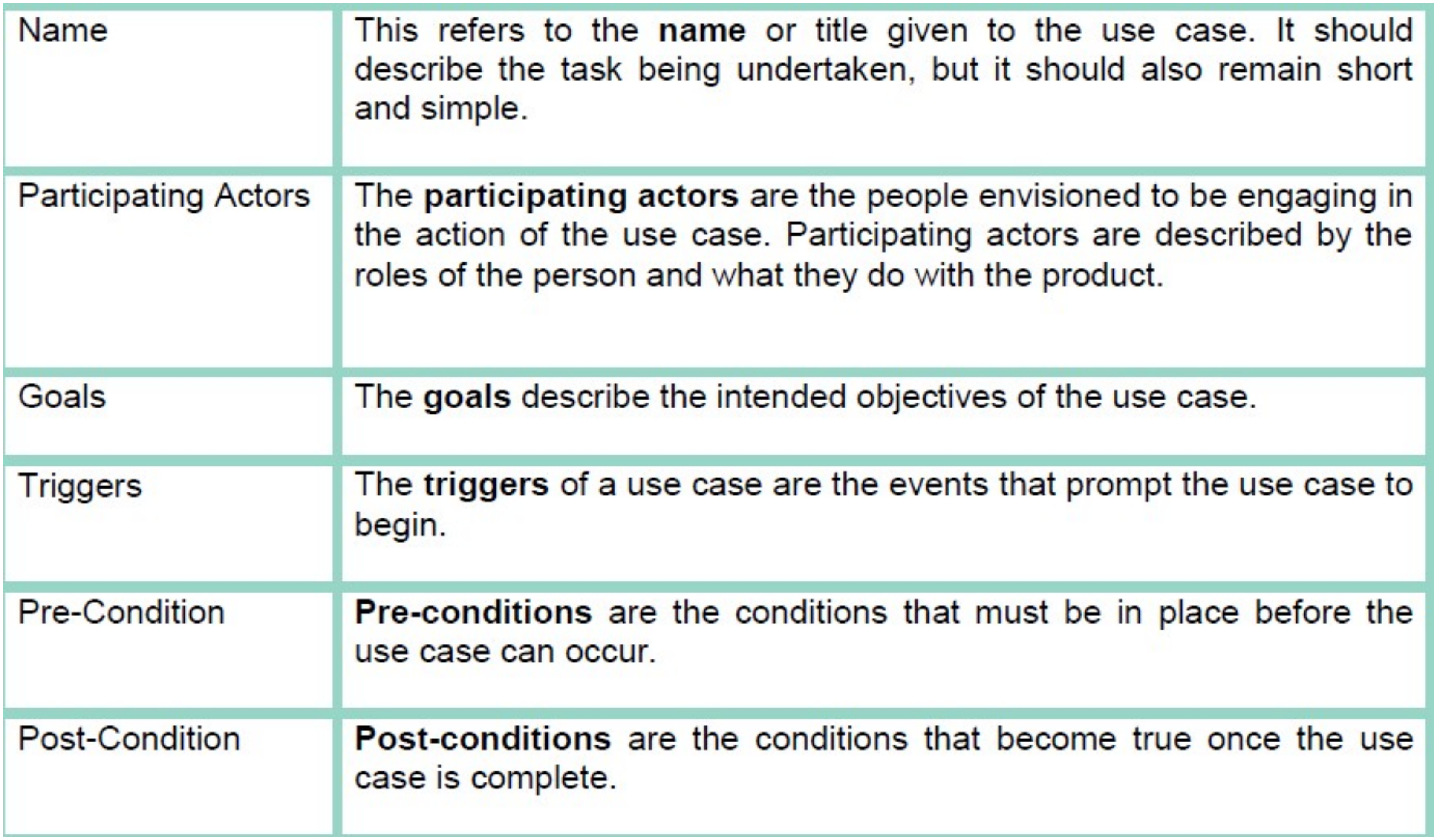
- Uses cases take place in particular environments to achieve particular goals. In other words, it’s a way of explaining a set of sequential interactions users might have with the features of a product.

- A good use case outlines the proposed task from the point of view of a participating actor (usually a user), and it should not require deep knowledge of technology to understand.

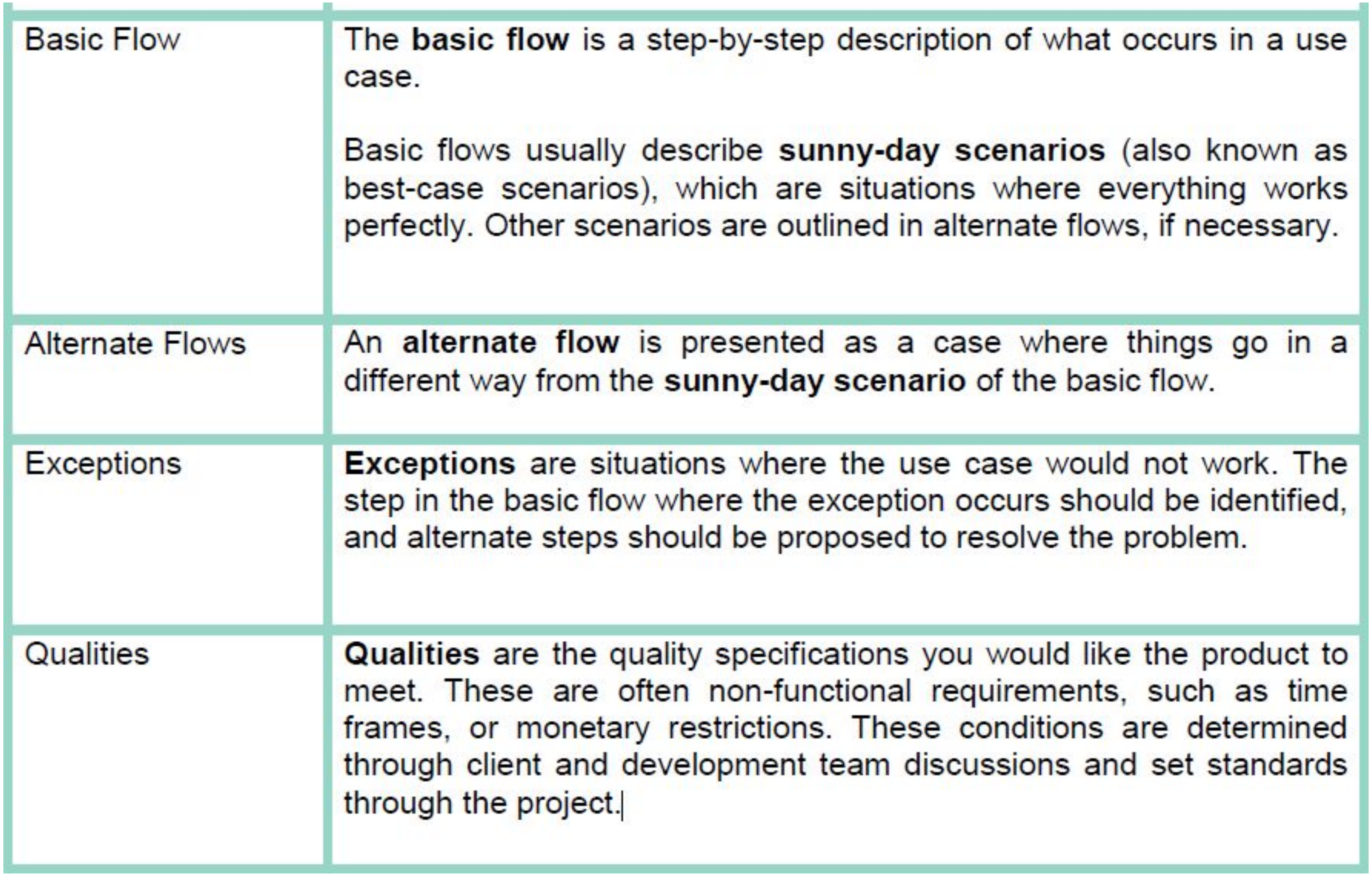
USE CASE FORMAT



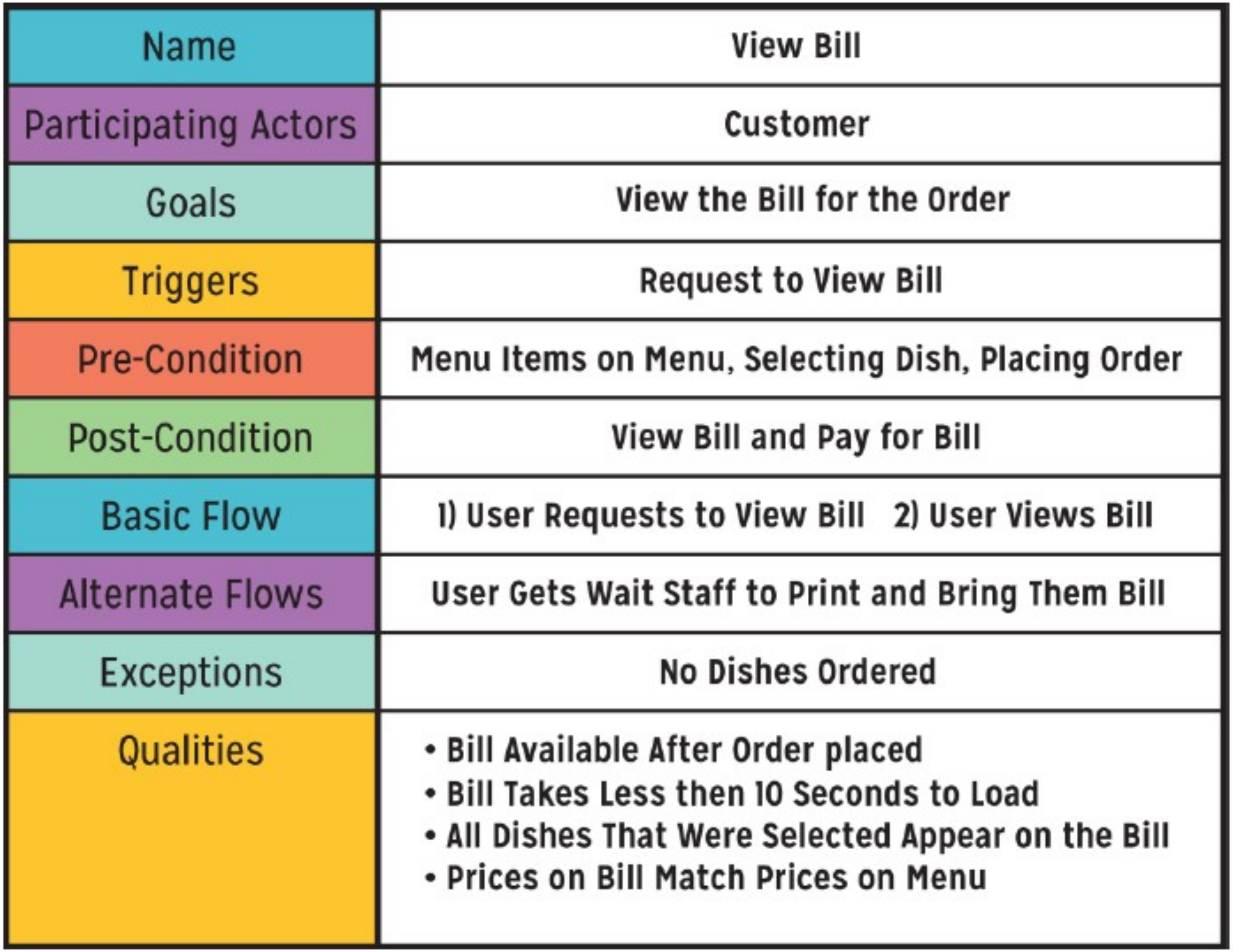
Name	
Participating Actors	
Goals	
Triggers	
Pre-Condition	
Post-Condition	
Basic Flow	
Alternate Flows	
Exceptions	
Qualities	



Name	This refers to the **name** or title given to the use case. It should describe the task being undertaken, but it should also remain short and simple.
Participating Actors	The **participating actors** are the people envisioned to be engaging in the action of the use case. Participating actors are described by the roles of the person and what they do with the product.
Goals	The **goals** describe the intended objectives of the use case.
Triggers	The **triggers** of a use case are the events that prompt the use case to begin.
Pre-Condition	**Pre-conditions** are the conditions that must be in place before the use case can occur.
Post-Condition	**Post-conditions** are the conditions that become true once the use case is complete.

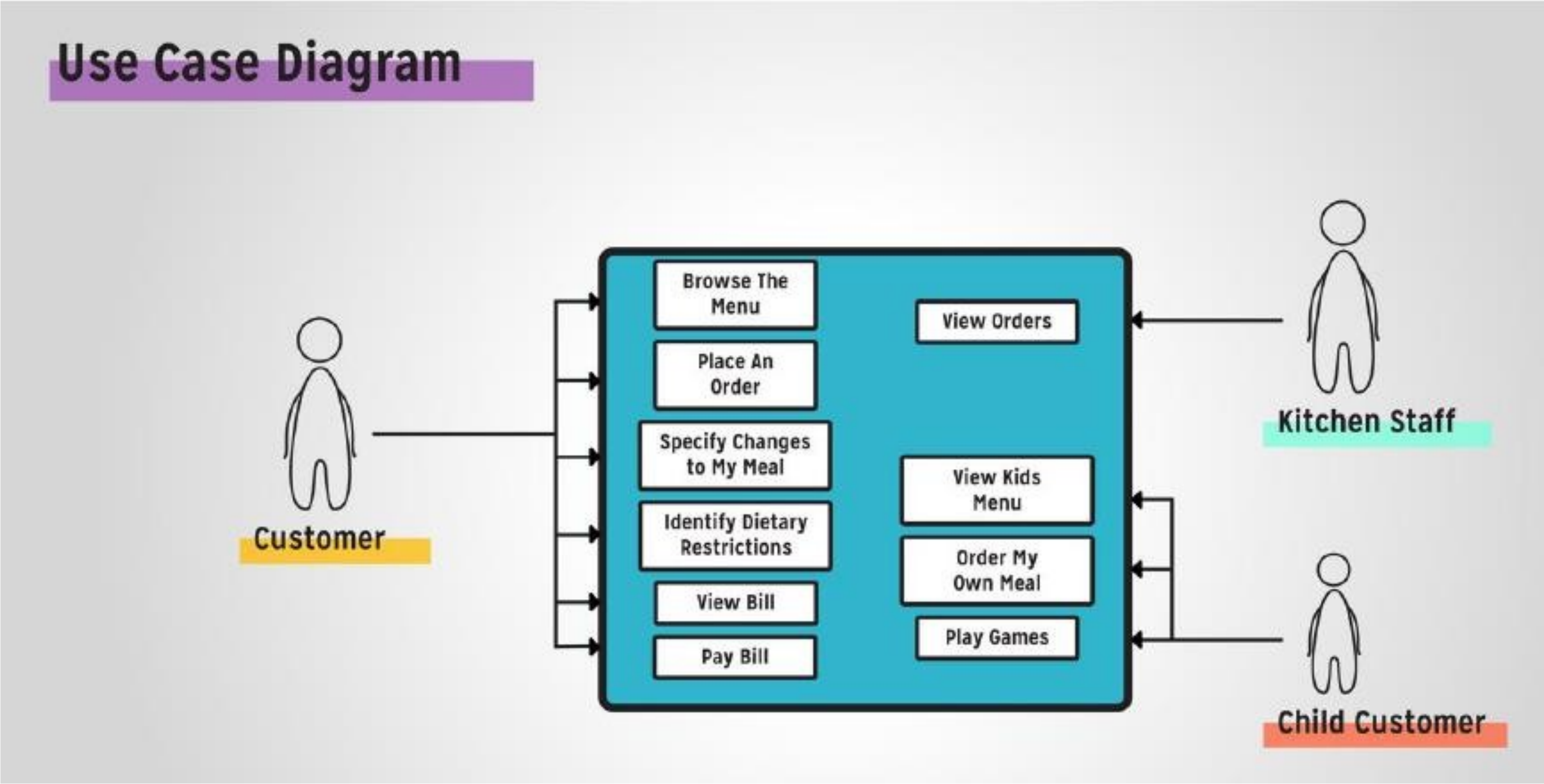


Basic Flow	The **basic flow** is a step-by-step description of what occurs in a use case. Basic flows usually describe **sunny-day scenarios** (also known as best-case scenarios), which are situations where everything works perfectly. Other scenarios are outlined in alternate flows, if necessary.
Alternate Flows	An **alternate flow** is presented as a case where things go in a different way from the **sunny-day scenario** of the basic flow.
Exceptions	**Exceptions** are situations where the use case would not work. The step in the basic flow where the exception occurs should be identified, and alternate steps should be proposed to resolve the problem.
Qualities	**Qualities** are the quality specifications you would like the product to meet. These are often non-functional requirements, such as time frames, or monetary restrictions. These conditions are determined through client and development team discussions and set standards through the project.



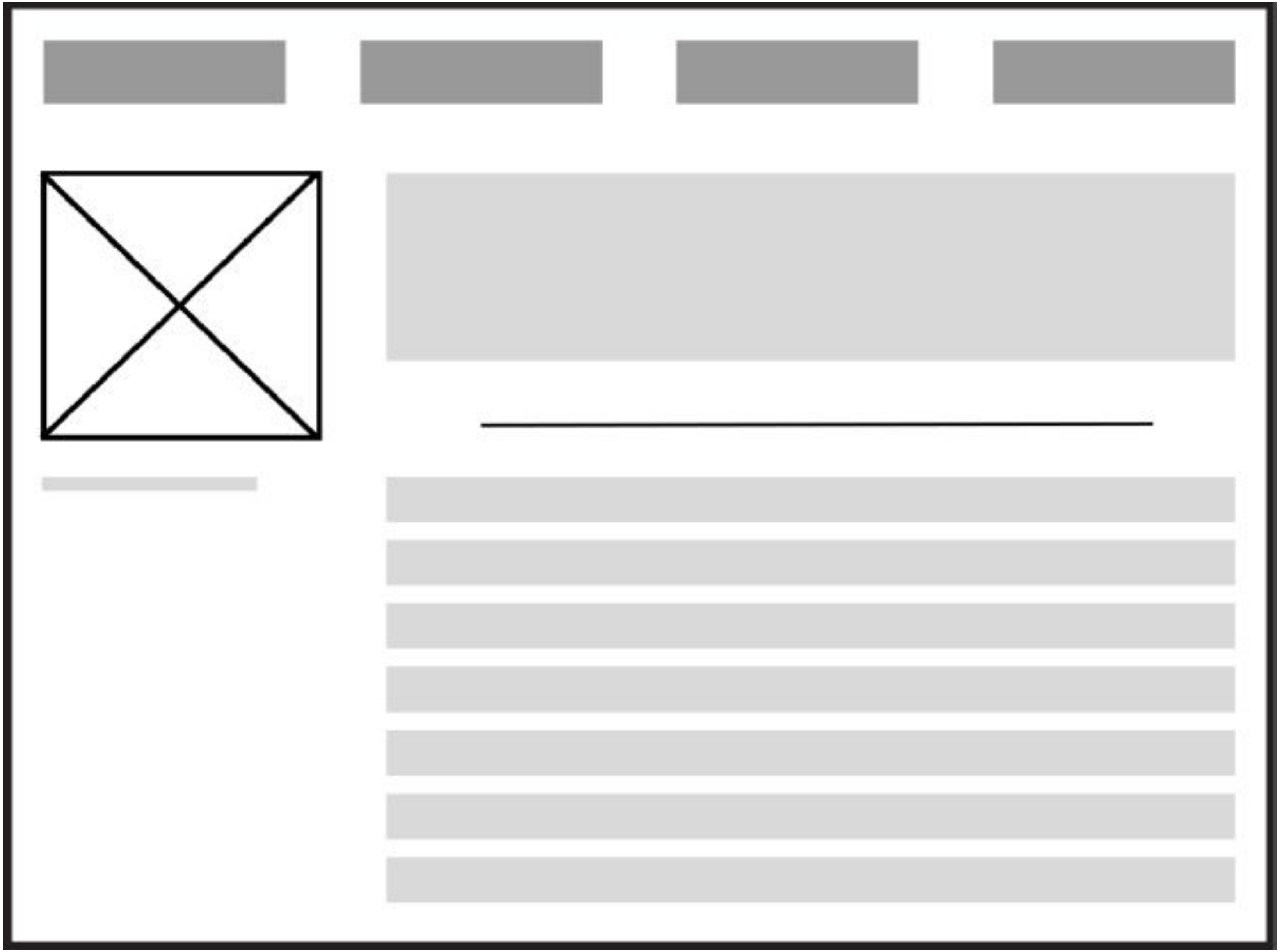
Name	View Bill
Participating Actors	Customer
Goals	View the Bill for the Order
Triggers	Request to View Bill
Pre-Condition	Menu Items on Menu, Selecting Dish, Placing Order
Post-Condition	View Bill and Pay for Bill
Basic Flow	1) User Requests to View Bill 2) User Views Bill
Alternate Flows	User Gets Wait Staff to Print and Bring Them Bill
Exceptions	No Dishes Ordered
Qualities	• Bill Available After Order placed • Bill Takes Less then 10 Seconds to Load • All Dishes That Were Selected Appear on the Bill • Prices on Bill Match Prices on Menu

USE CASE DIAGRAM



WIREFRAMES

One of the most important techniques of product development is the use of **wireframes**. A wireframe, also known as a mock-up, can be thought of as a kind of early blueprint.



USAGE OF WIRE FRAMES

- Wireframes are used for many purposes. These include:
  - getting an idea for what will be developed
  - demonstrating ideas to clients or users and encouraging their feedback and involvement
  - communication among the development team

USAGE OF WIRE FRAMES

- helping the client or users communicate with the software product manager and team (some people may find it easier to sketch out their ideas to describe them)

WHAT WIREFRAMES ARE NOT

- Although a visual representation, a key aspect of wireframes is their simplicity. Wireframes are not meant to be a demo of the interface of a product but rather outline basic functionalities and end-user tasks.

- This means that wireframes do not use colours or images. In fact, it is important that detailed design features do not creep in to wireframes.

- Instead, wireframes may suggest things like where buttons, text fields, and images are placed.The elements displayed should theoretically allow a user to do a task with the product.

- As in a blueprint for a house, for example, details such as wall colours or lighting fixtures are not outlined. Later, detailed user interface (UI) design can develop further

- TIP:There are many tools to help develop wireframes! You can always draw images by hand and then scan them, but there are many drawing tools on the computer that can help you create wireframes.
- Some examples are:

- Pencil (Open source)
- JustInMind (more sophisticated Recommended)
- Google Docs
- Microsoft Powerpoint
- Balsamiq (https://balsamiq.com/)
- OmniGraffle (paid app)
- Wirify

- Some examples of wireframes can be found at http://wireframes.tumblr.com/

- Class practice
- Write use case of Login and draw the wireframes.