### **Goals of today**

* Create product backlog  
  Product Backlog (systematically represent the needs of the client)
  + Contains all "features"
  + The features are prioritised by various factors:
    - Risk/impact
    - Added value
    - Required data
* → Verify if the interpretation is indeed correct (either do this today or next week)  
  → Propose own ideas, and verify their priority to allow for proper selection for **sprint**
* Setup communication rules for the entire team + TA  
  → In our Javadoc we can specify who wrote that code  
  → Use of Whatsapp or discord?
* Make a planning for the upcoming weeks  
  → Define specific milestones per week  
  → Spread work out evenly  
  Distribute each task based on *effort* for each person and set their deadline
* Create a clear definition of "done"

### **General Goals**

* Make sure to prepare your meetings, make an agenda of what you want to discuss and take the lead.
* Preferably make an agenda and share it with your TA prior to the meeting.
* Document each decisions you make  
  → Helps for the report we need to write  
  → Helps for peer feedback

### 

### **Campus Management Software Details**

Aim:

→ Things related to life/studying on TU Delft campus

Build:

→ A backend that exposes an API

(Spring MVC Framework)

→ A client application that consumes this API

(OpenFX Framework)

* All data stored on a DB

**Questions for CTA**

* Main aim/goal/purpose of the software application
* Main target audience of the software application (ie. who will be using it: student, teacher, how many expected users, etc)
* List of all features that you would expect for a specific role (administrator/general user) + the features significance
* What views are expected (website, desktop application, mobile application)

### 

### **Software notes**

Java on its own is both a programming language and a platform.

→ The *Java programming language* is a high-level object-oriented language that has a particular syntax and style.

→ A *Java platform* is a particular environment in which Java programming language applications run.

Java SE vs Java EE

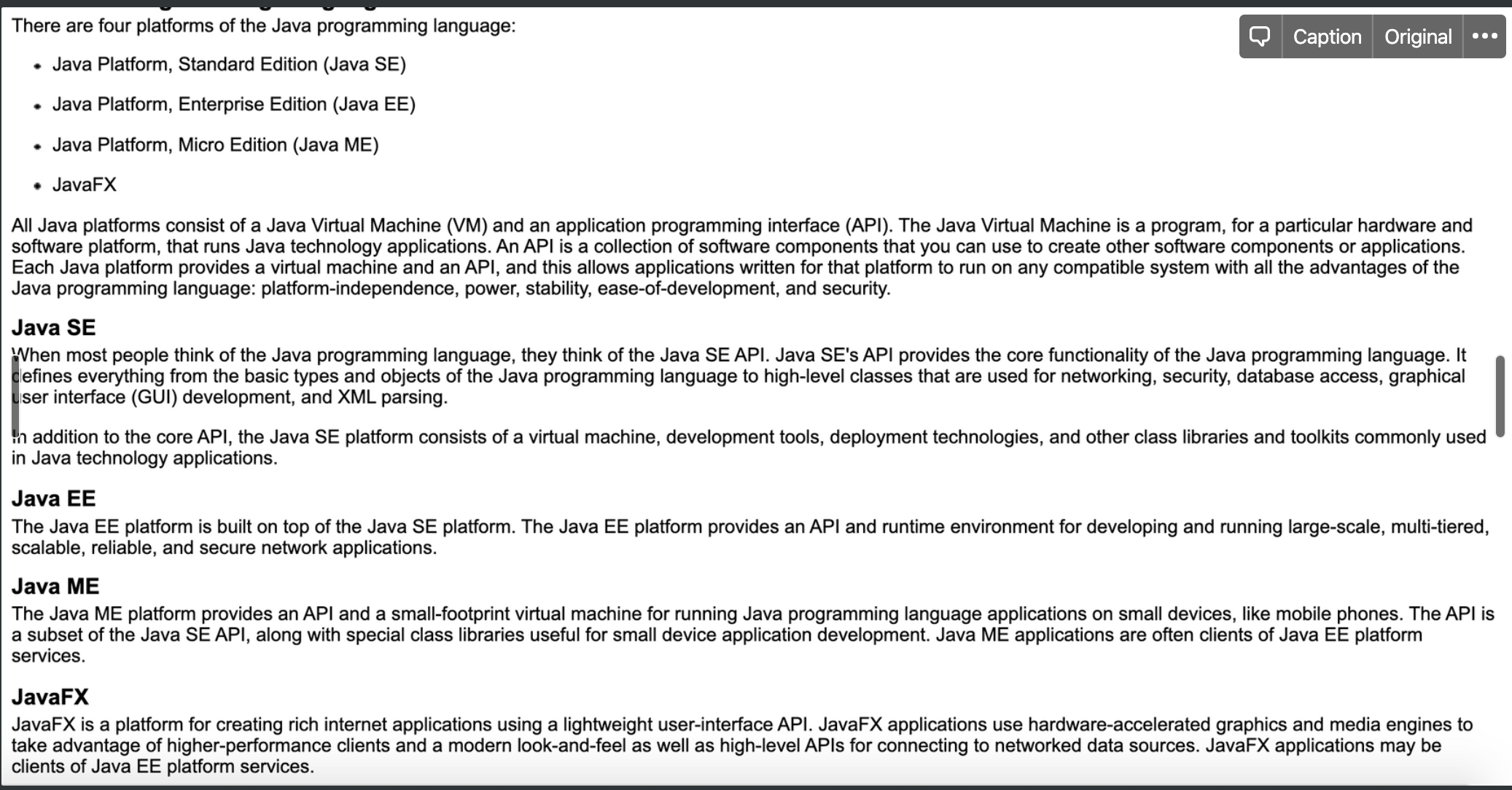
→ The Java EE platform is built on top of the Java SE platform

→ The Java EE platform provides an API and runtime environment for developing and running large-scale, multi-tiered, scalable, reliable, and secure network applications.

→ The Java Platform, Enterprise Edition (Java EE) is a collection of Java APIs owned by Oracle that software developers can use to write server-side applications.

→ Java EE provides **transaction support.**

→ The Java EE product provider is typically an application-server, web-server, or database-system vendor who provides classes that implement the interfaces defined in the specifications. These vendors compete on implementations of the Java EE specifications.



So essentially, we use Java EE to create the server side of our application, and we use JavaFX to create the client side.

**The Spring framework**

<https://www.edureka.co/blog/java-frameworks/>

<https://www.youtube.com/watch?v=PjyVXs0_3H8>

<https://www.youtube.com/watch?v=gq4S-ovWVlM>

<https://spring.io/guides/gs/rest-service/>

<https://www.edureka.co/blog/spring-mvc-tutorial/>

<https://www.edureka.co/blog/videos/introduction-to-spring-webmvc-spring-with-bigdata/>

<https://www.youtube.com/watch?v=g2b-NbR48Jo>

<https://www.youtube.com/watch?v=If1Lw4pLLEo>

<https://www.youtube.com/watch?v=rMLP-NEPgnM>