What is a Product Backlog?

• A term introduced by the Scrum methodology
• A place where all the functional (and non-functional) “requirements” are kept
• It may be an Excel file, or a GitHub artifact (like in our case)
• In agile methodologies, the term “requirement” is replaced with a “user story”
In the next slides we will learn some principles for writing good user stories. They are taken from the book “Planning Extreme Programming”

Stories must be understandable to the customer

- It’s no good making the requirements so difficult to write and organize that you need years of training in requirements engineering to be able to understand them
- The language for a story is plain English (or whatever your local language is)
Write user stories on index cards

- Cards keep stories concise and also make them easy to manipulate during planning sessions.
- If you're determined to put stories into a computer, do so in such a way that you can easily print them out on cards using standard printer card stock.

When you plan the Backlog use index cards! However you should eventually add them to GitHub.

The shorter the story the better

- The story represents a concept and is not a detailed specification.
- A user story is nothing more than an agreement that the customer and developers will talk together about a feature.
- The best user story is a sentence or two that describes something important to the customer.

For example: “The system should check the spelling of all words entered in the comments field.”
How big a story should be?

• One of the hardest things about stories is how big to make them
• Stories need to be of a size that you can build a few of them in each iteration
• This size gives you the ability to steer by shifting stories between iterations
• It also means that the developers should be able to estimate how long it will take to do a story
• If they can't, it usually means the story needs to be broken down into smaller parts

Stories should be independent of each other

• This allows us the freedom to build them in any order
• This is, of course, impossible.
• But in practice we find that if we pretend it is possible, most of the time we get away with it
• We do get dependencies, but rarely will they cause a problem with the planning process
Each story must be testable

• When the story gets built it will be important to be able to know that it works
• So each story must be testable
• You don't have to write the test right now, but you should be able to figure out how to test whether the story is there or not

Additional course guidelines

• The Product Backlog should contain user stories and not development tasks
• User stories are short descriptions of user/customer visible functionality
• Development tasks (that you should not define at this stage), are derived from the user stories and are meant for developers
• User stories should be described in a language that the user understands. Do not use words such as "Fragment" or "Intent"
• Ideally, user stories should not describe the structure of the user interface but only the desired functionality
• User stories should be short and concise
Creating a Backlog in GitHub

• You should create a **Project** in your GitHub repository called "ProductBacklog"
• Within the Project, you should create the stories as **notes**
• Creating them as issues is less recommended but possible
• The stories should be sorted by importance
• All stories should be located in a column called **ALL**
• During the planning meeting we will move stories to new columns: **Sprint1** & **Sprint2**
Tip: push your initial project now

• The Backlog submission includes creating a README file
• It is easier to “push” to an empty repository than to push to a non-empty one (with a README file)
• Following slides explain how to do so

Create an Android project with .git directory

• Create an Android Studio project and remember the project’s directory path e.g., 
  C:\Users\omishali.TD-CSF\StudioProjects\HelloAndroid
• Change to the directory and execute *git init*

Now we need to turn the project’s directory into a git repository

Eventually, a .git directory is created within the project’s directory
Update the .gitignore file

• Note that Android Studio created the project with a default .gitignore file
• We have found this default file to be incomplete, and thus replaced it with the file listed in this stackoverflow thread (first answer) or in this link
• We suggest you to do the same change in your project

Important your project may have two .gitignore files. Leave the one within the “app” directory as it is, and modify the second one (within a directory having the project’s name)

“Push” the project from the git shell

• Next, we will use the git shell to push the project’s code to the repository we have in GitHub
• Open a git shell that points to your Android project, and executes the following commands:

  - git add . # note the period.
  - git commit -m “first commit”
  - git remote add origin https://github.com/Technion236503/hello-android.git
  - git push -u origin master

Congratulations! You have just committed the project to GitHub!

You may also open a git shell from GitHub Desktop
Git clone – get a local copy of a GitHub repo

- For other team members to work on the project, they should **clone** (copy) the GitHub repository
- Point the *Git Shell* to a directory where you want the repository to be cloned
- Copy from GitHub the repository URL
- Execute the command `git clone <url>`
- Now you have a local copy of the team’s GitHub repository
- Open the project in Android Studio

This operation could also be done using *GitHub Desktop*

Git development workflows

- There are several possible development workflows for a git team
- **This great article** compares 4 workflows:
  - Centralized Workflow
  - Feature Branch Workflow
  - Gitflow Workflow
  - Forking Workflow
- We believe that the first centralized workflow is suitable for your project and you should read at least that part
- You are encouraged to read other parts as well