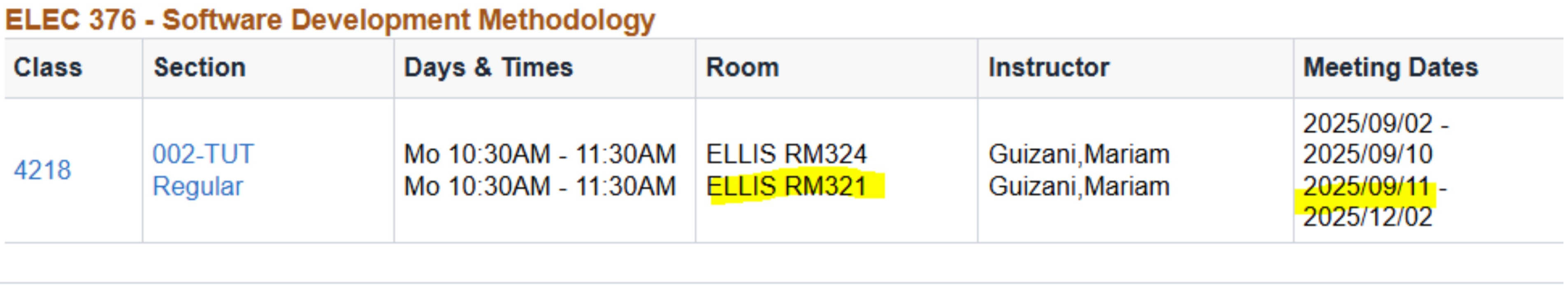
Scrum, Arrays and Vectors

ELEC 376 (week 2)

Change of tutorial classroom

We moved tutorial classroom from ELLIS RM324 to **ELLIS RM 321**



ELEC 376 - Software Development Methodology

Class	Section	Days & Times	Room	Instructor	Meeting Dates
4218	002-TUT Regular	Mo 10:30AM - 11:30AM Mo 10:30AM - 11:30AM	ELLIS RM324 ELLIS RM321	Guizani,Mariam Guizani,Mariam	2025/09/02 - 2025/09/10 2025/09/11 - 2025/12/02

Requirement Analysis Document (Due Week 3, Sep 20th)

Live on onQ

Mandatory sections:

- Executive summary
- Background and history
- Purpose of the systems
- Scope of the system
- Objective and success criteria
- Definition and acronyms
- Description of the system
- e.g., program flow using a flow diagram, textual description, or both
- Functional requirements (using user stories)
- Non-functional requirements (using user stories)
- high-fidelity prototyping (e.g., you can use Figma)
- Assumptions and constraints
- Roles of the team (assuming everyone is a developer first)
- Quality of the writing and rationale statements added when needed
- <u>A table detailing who wrote what (Individual Contribution)</u>
- Note: Please also refer to the two RAD examples shared on onQ.

GitLab and repo access

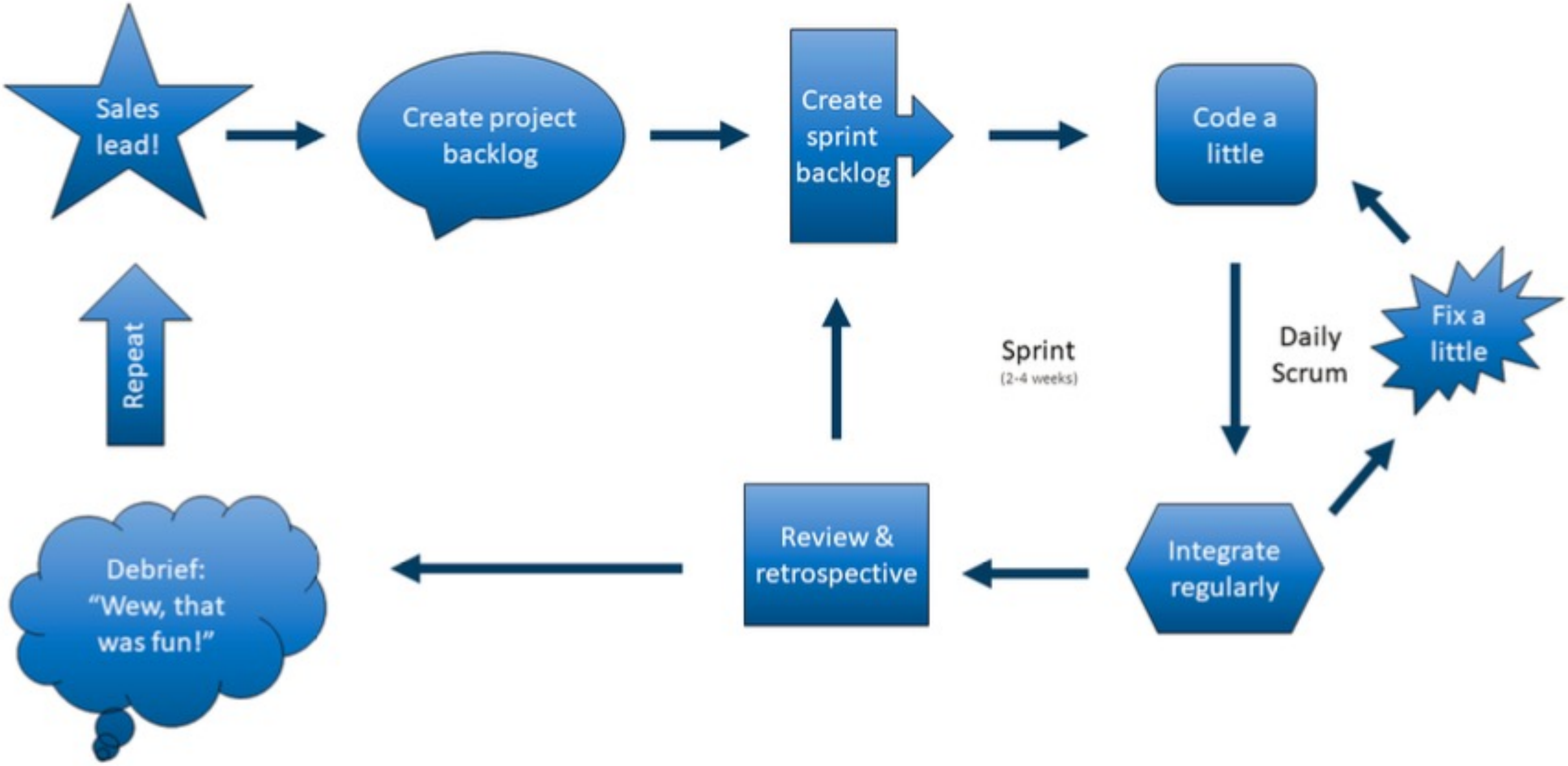
Link to access Group Project Repo:

https://code.smithengineering.queensu.ca/elec376/project_name/

Remember front loading the coding in scrum

Also, your first sprint report is coming up in Week 5

Scrum Overview



Scrum project

A project: some **initial planning** plus several sprints

Initial planning

- Who is the customer?
- Who are the users?
- Who is the team?
- **What is the initial project backlog?**

Requirement Analysis Document (RAD)

Scrum project

A project: some initial planning plus **several sprints**

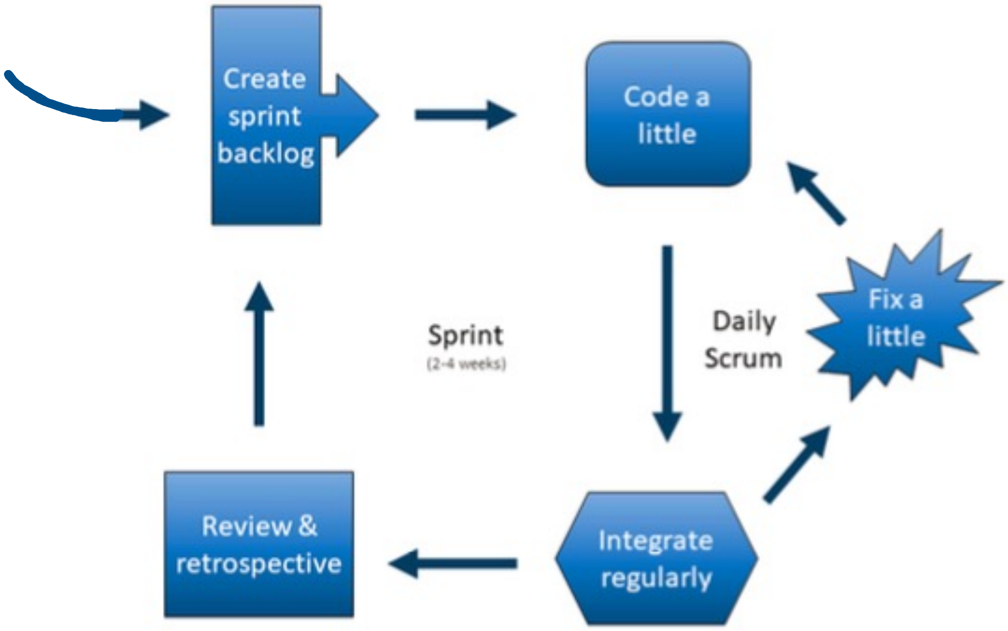
- Each sprint is **2-4 weeks** long
  - Sprint backlog
  - The development team implements
  - Daily meetings called ”daily scrum” (15 min)
  - Ends with **Sprint Review** and **Sprint Retrospective**

Sprint (Sprint one Starts next week)

Filling the sprint backlog

- Select **stories**, tasks **based on estimates**

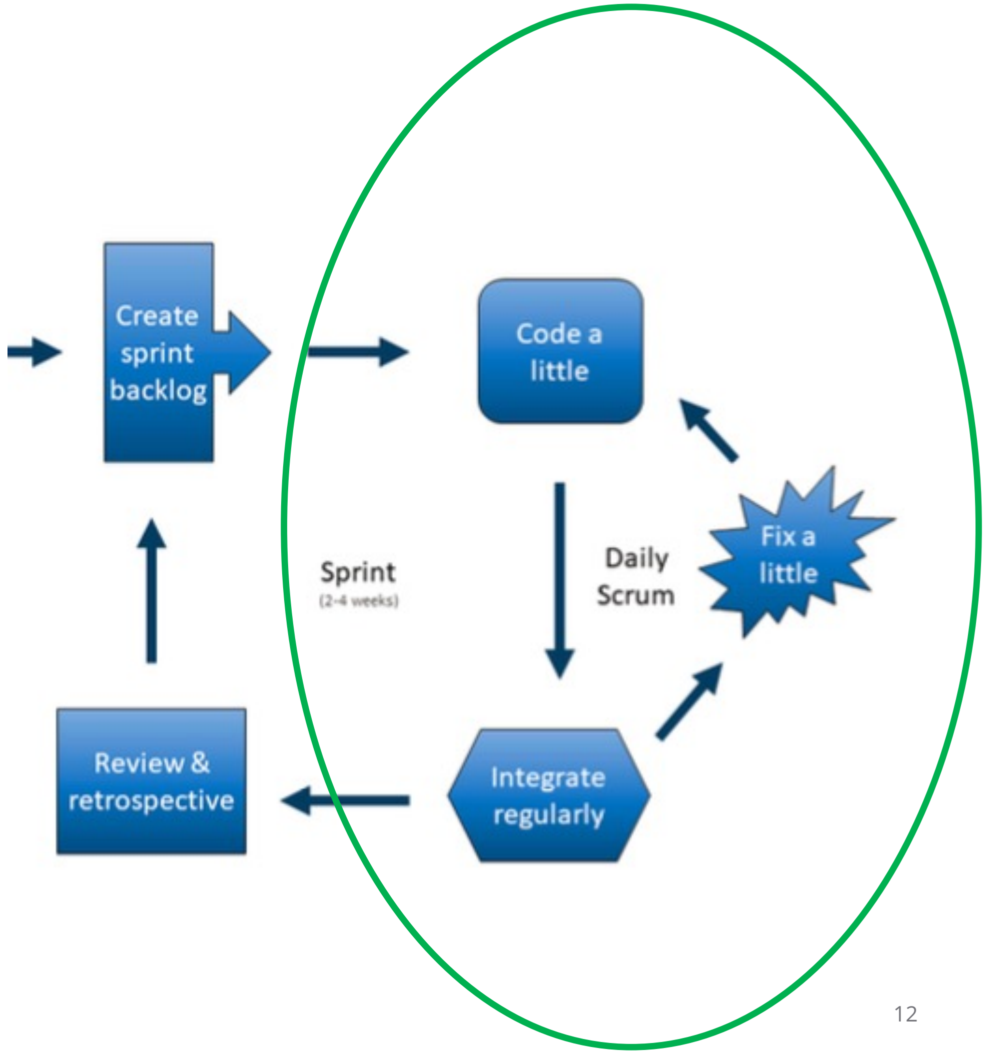
- Should have a single **unifying** Sprint **goal**



Sprint

Code, test, integrate, and iterate

- Aim for each feature to be 100% done
- Half-finished features count for nothing



Sprint

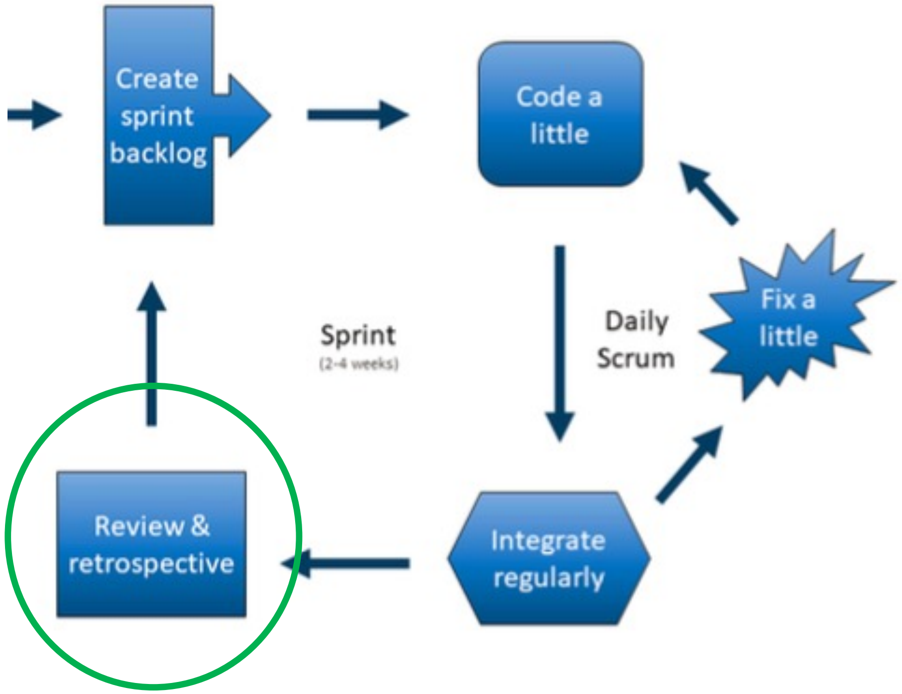
Sprint review

- Presentation of a demo to users
- Software should be ready for release

Sprint retrospective

- Debrief to find ways of improving

Week 3



Week 5

Time management

Best practices:

All meetings conducted in person

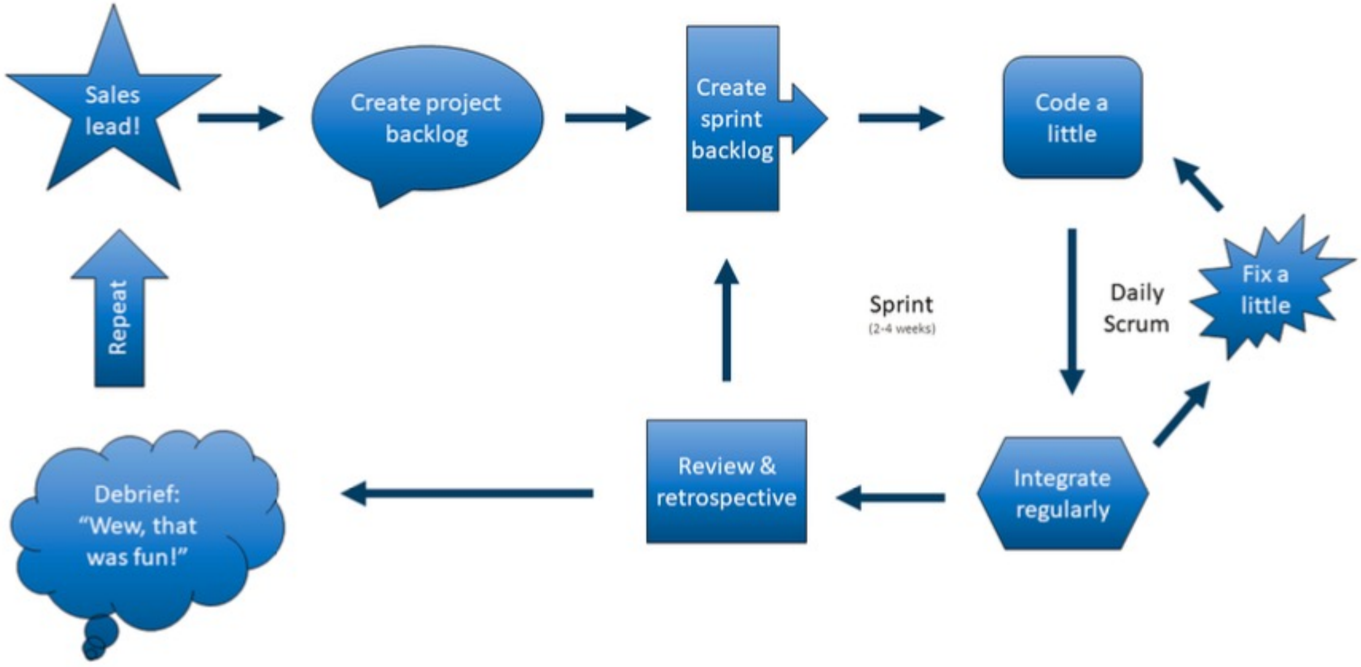
Implementation possible remotely

Scrum allocates time meticulously for everyone:

- **Initial project planning:** At most, one day
- **Sprint planning:** Limited to a few hours
- **Sprint duration:** Maximum of one month
- **Daily Scrum:** Lasts no more than 15 minutes
- **Sprint Review:** Takes a few hours at most
- **Sprint Retrospective:** Also a few hours at most

Scrum project roles

- Product Owner
- Scrum Master
- **Development Team**



Scrum project roles

Product Owner

- Responsible for communicating with **customers** to understand requirements
- Holds ownership of the **Product Backlog**
  - It is ok to share this responsibility with teammates.
  - But in the end, the Product Owner is responsible.
- May handle recording the Sprint Backlog chosen by the team
- Going to lead the effort to set your project's direction
- Frequently leads the **Sprint Review** with the customer

Scrum project roles

Scrum Master

- Not the product owner
- Responsible for ensuring proper implementation of Scrum practices
- Documenting scrum practice
- Maintains the updated **Sprint Backlog**
- Often leads the **Sprint Retrospective**

Scrum project roles

Development Team

- Responsible for creating the software
- Occasionally includes the Product Owner and Scrum Master
  - In your projects Scrum Master and Product Owner will also be part of the development team

- No differentiation between design/code/integration/QA

- Everyone involved in every aspect

By now you should

- Identify your Product Owner and Scrum Master
- A **Product Backlog**

Initial planning

Product Backlog:

Collection of user stories

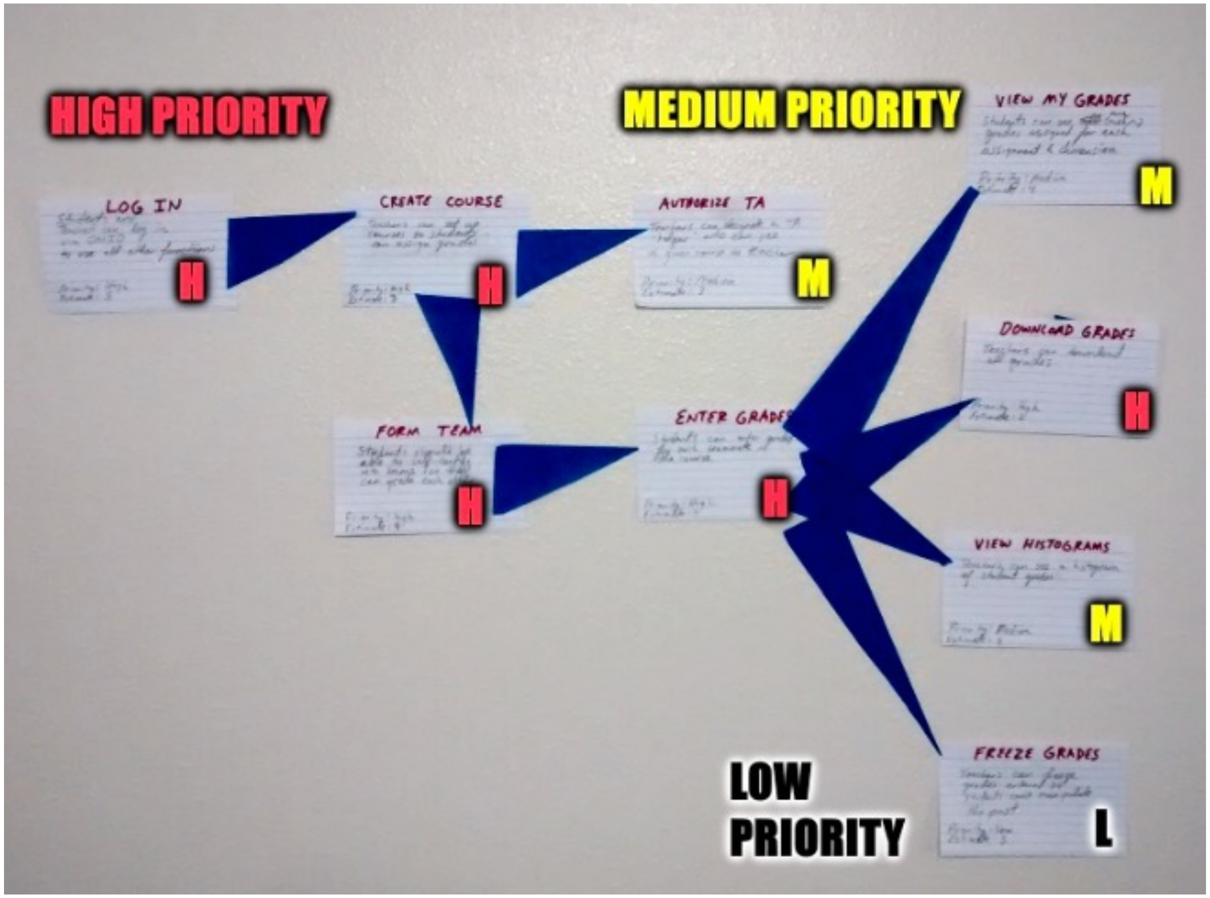
Basic information

- Name
- One-sentence summary
- Notes & acceptance criteria

And, based on discussions:

- Effort estimate (person-hours)
- Priority (essential/high/medium/low)

How to assign priorities



Main consideration:

- value to users

Secondary consideration:

- logical dependencies

Each story has tasks

Example: Log in story has several tasks...

- Create code to check if user is already logged in, or is logging in (authenticate)
- Create code to redirect to login page
- Create the login page itself

Complete Initial planning

Discuss tools, platforms, etc.

- Then, make a choice
- If anybody will need help getting up to speed, make a note of it
- Plan to work in pairs for a few weeks if needed

Based on your choice of tools...

- Create a "setup" story that consists of setting up tools

Break down each story into a set of tasks

- And, for each task, estimate the hours required

Setup story

- "setup" story for your project

  - Setting up the environment

  - Create a user database

  - Create "Hello, World" test to verify setup

Complete initial planning

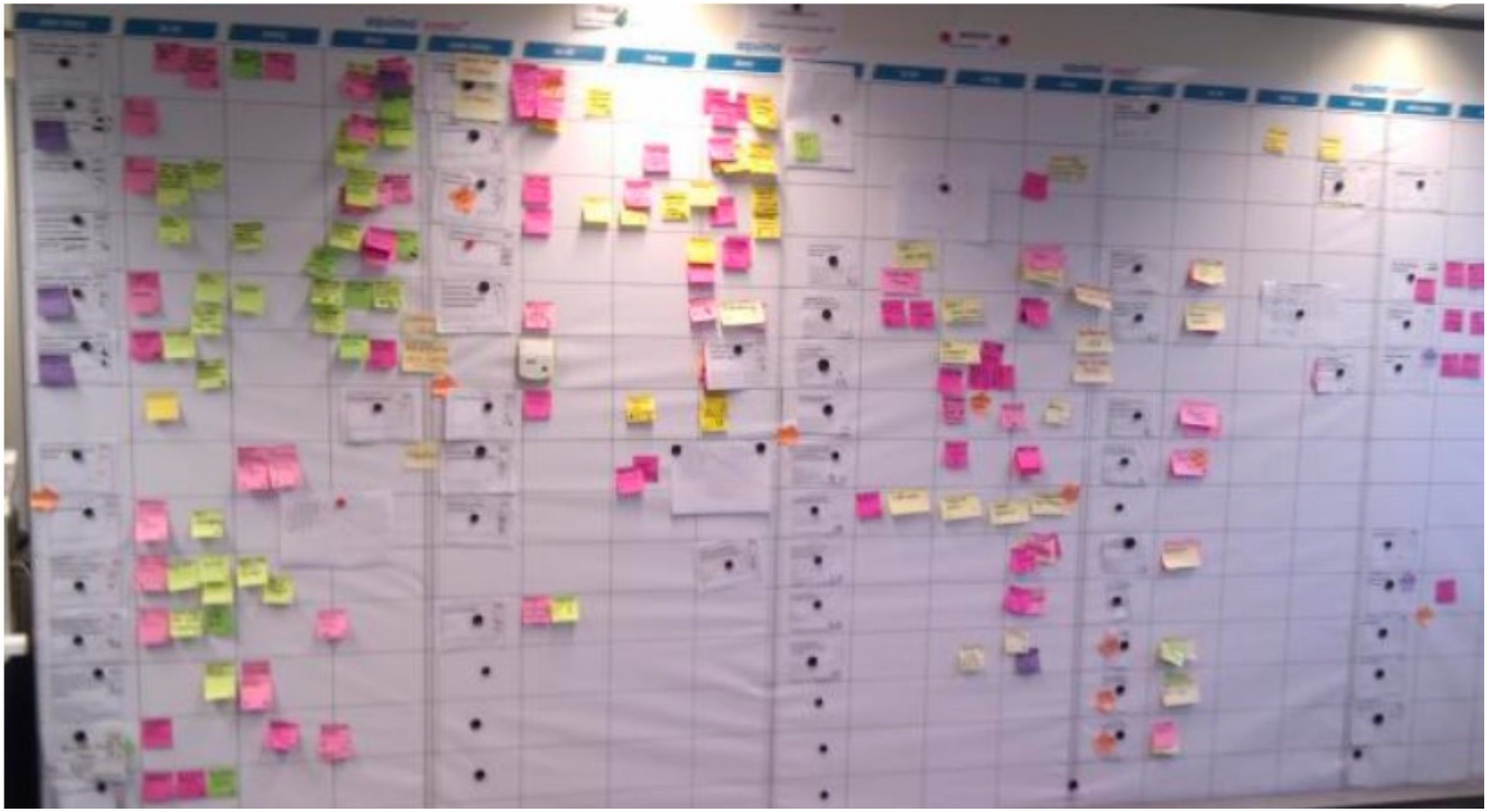
Write these tasks on post-it notes (or virtual)

For each 3x5 card, you should have several post-its

- Write your estimates on the post-it notes
- Write the sum on 3x5

- Product Owner should collect everything:
- **Product Backlog** complete!

Product backlog can look like



On performing scrum

Increment

Each sprint produces next increment

This increment is a fully-releasable software system

- Maybe you will not choose to release it
- But you could

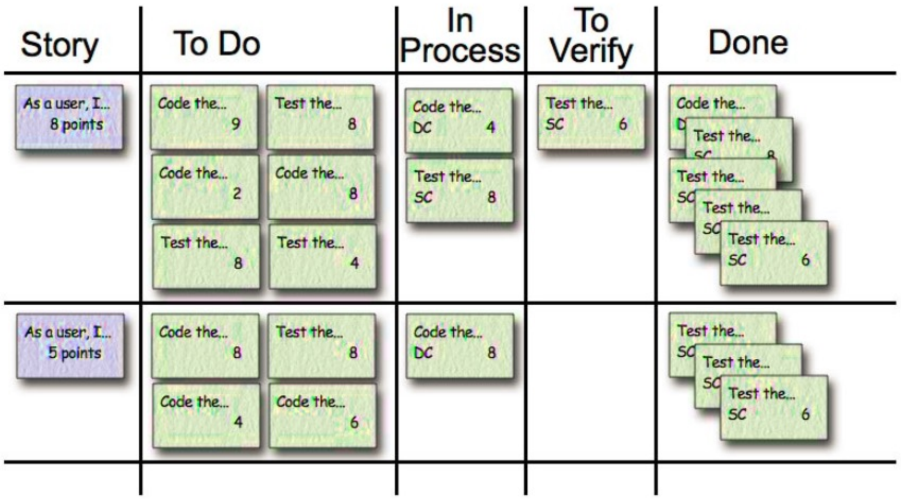
No half-finished features

On performing scrum

Sprint backlog

- Team must plan a sprint
- Select user stories that, together, can be a software release
- 3-week sprint
- The first sprint usually has lots of unexpected problems, so it is ok if you want to be conservative and plan **at only 75% of theoretical capacity.**

On performing scrum



Story	To Do	In Process	To Verify	Done
As a user, I... 8 points	Code the... 9 Test the... 8 Code the... 2 Code the... 8 Test the... 8 Test the... 4	Code the... DC 4 Test the... SC 8	Test the... SC 6	Code the... D Test the... SC 8 Test the... SC Test the... SC Test the... SC 6
As a user, I... 5 points	Code the... 8 Test the... 8 Code the... 4 Code the... 6	Code the... DC 8		Test the... SC Test the... SC Test the... SC 6

Sprint backlog

Absolutely necessary for team to discuss

- What time they can commit
- What challenges different stories present
- What will be done to overcome challenges
- Every single member of the team must be fully engaged.

On performing scrum

Daily scrum

Sometimes called a "stand up" meeting

- Everybody stands up,
- everybody speaks,
- everybody pays attention
- Only 15 minutes long. Guaranteed.

Scrum Master has each person state only three things:

- **What tasks did you accomplish since last meeting?**
- **What tasks will you accomplish before next meeting?**
- **What obstacles are in the way?**

Scrum Master must make notes about obstacles

Do not discuss the obstacles in detail during the meeting

On performing scrum

Scrum Master job after the meeting

If somebody has run out of tasks...

- Help them choose the next one (based on priority)
- Get the Product Owner to help choose if needed

If somebody has an obstacle...

- It is your job to help your team succeed
- Give help, or get help
- Maybe pair off with a partner

If somebody hasn't been doing Scrum right...

- Get them to pair off with another teammate for a day

On performing scrum

When you fall behind

Finish user stories before starting new ones

- Half-finished stories count for nothing

Get the highest-priority user stories first

- **Product Owner's** job to keep the team focused

Don't let your emotions control you

- Remember you're not alone
- Work as a team; be generous with your time; ask for help

On performing scrum

When you really fall behind

Not the case of the class Sprints

When you're really far behind...

In the real world, sprints sometimes get canceled

- Usually for business reasons
- Easier to combine 2 releases than to drag one out

On performing scrum

Spring review

Product Owner's responsibility to meet with users (TA)

- Obtain evidence that they did use your system
- Obtain feedback about strengths and weaknesses
- Development team may attend, but not necessary (rare)

May stretch over several days in a large project

Often only involves a demo, rather than letting users actually try

- Stakeholders watching, able to contribute feedback

Use information to update Product Backlog

- Get a few teammates to help break work into tasks, assign estimates

On performing scrum

Sprint retrospective

Time-boxed at 3 hours

Scrum Master leads discussion

Every single development team member must answer

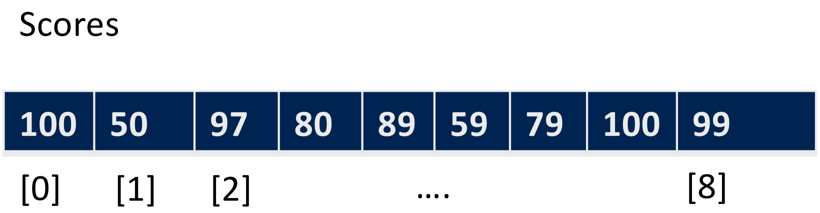
- What changes are needed...
- Work with Product Owner to update estimates

Arrays and Vectors

ELEC 376 (week 2)

Arrays

- **Fixed size**
- Same type elements
- Stored in memory contiguously
- Can access individual element using their position
  - First element at index: 0
  - Last element at index size -1
- Always initialize your array
- **No out of bounds checking**



Scores

100	50	97	80	89	59	79	100	99
[0]	[1]	[2]		….				[8]

Arrays declaration

- `Element_Type array_name [number of element]`

```
double lo_temps [4];
```

- `Element_Type array_name [number of element] {init list}`

```
double hi_temps [6] {45.5, 37.9,
35.1, 40.8};
```

?

```
double lo_temps [6] {0.0};
```

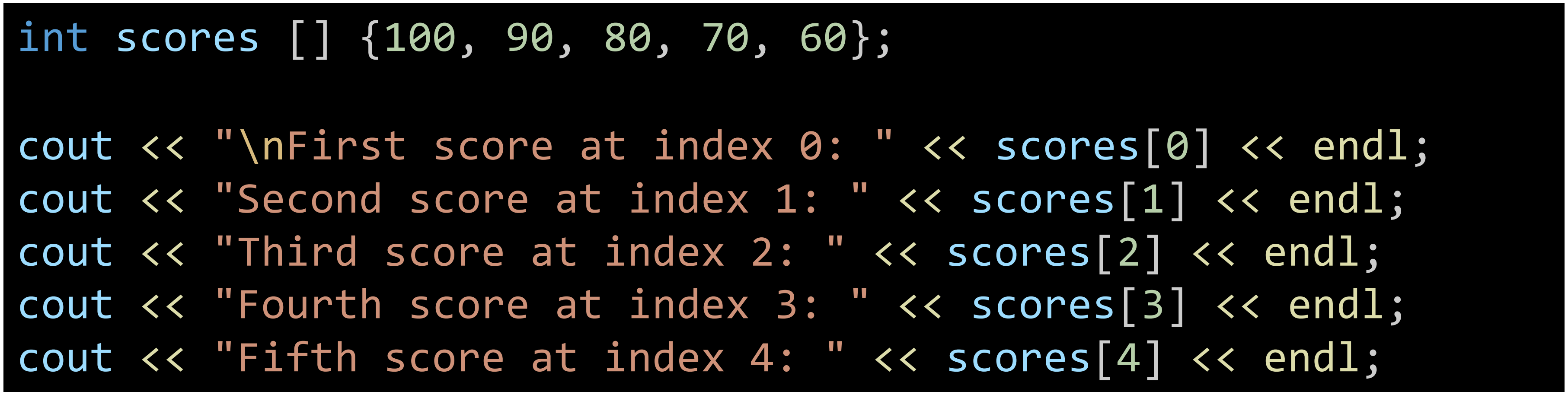
?

- Element_Type array_name [] {init list} // size?

```
char vowels[] {'a' ,'e', 'i', 'o', 'u' };
```

Accessing array elements

- array_name [index]



```
int scores [] {100, 90, 80, 70, 60};

cout << "\nFirst score at index 0: " << scores[0] << endl;
cout << "Second score at index 1: " << scores[1] << endl;
cout << "Third score at index 2: " << scores[2] << endl;
cout << "Fourth score at index 3: " << scores[3] << endl;
cout << "Fifth score at index 4: " << scores[4] << endl;
```

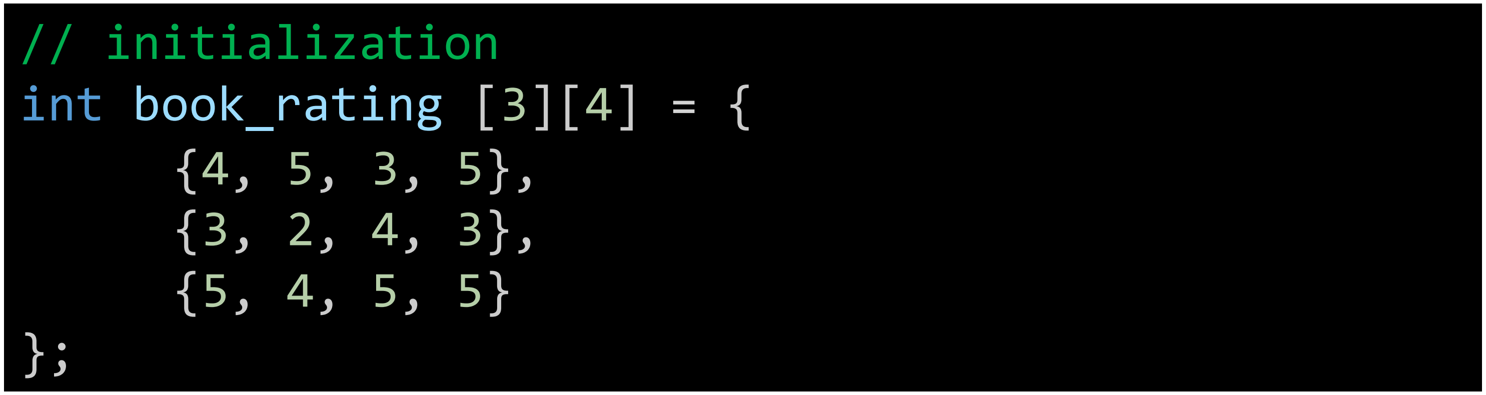
Accessing array elements

- Array name is the location of the first element of the array
- [index] is the offset from the start of the array
- Notes: no bound checking is performed with arrays! → potential garbage data

Multi-dimensional Arrays

- `Element_type array_name [size_dim1] [size_dim2]`
  - `int book_rating [3] [4]; // garbage data no init`

row column



```
// initialization
int book_rating [3][4] = {
      {4, 5, 3, 5},
      {3, 2, 4, 3},
      {5, 4, 5, 5}
};
```

Scenario

- Suppose □want to store the score of a player in an online game
- Players can join and drop out during a given game
- We have no way of knowing how many players are going to be at a given game

- What can we do?

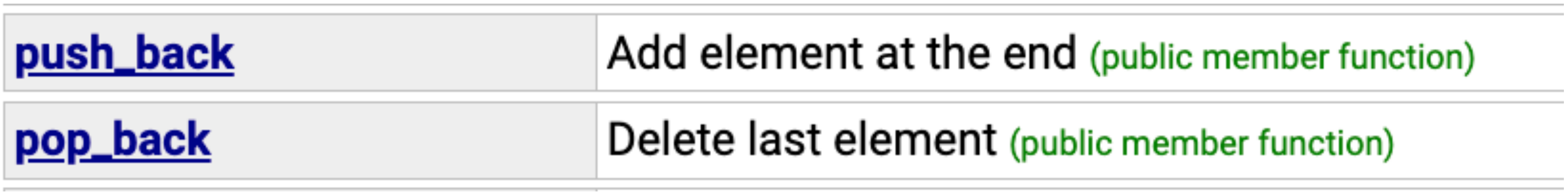
Vectors

Part of the C++ Standard Template Library (STL)

Dynamic in size: can grow or shrink in size at runtime

Lots of functions available: sort, reverse, find, ….

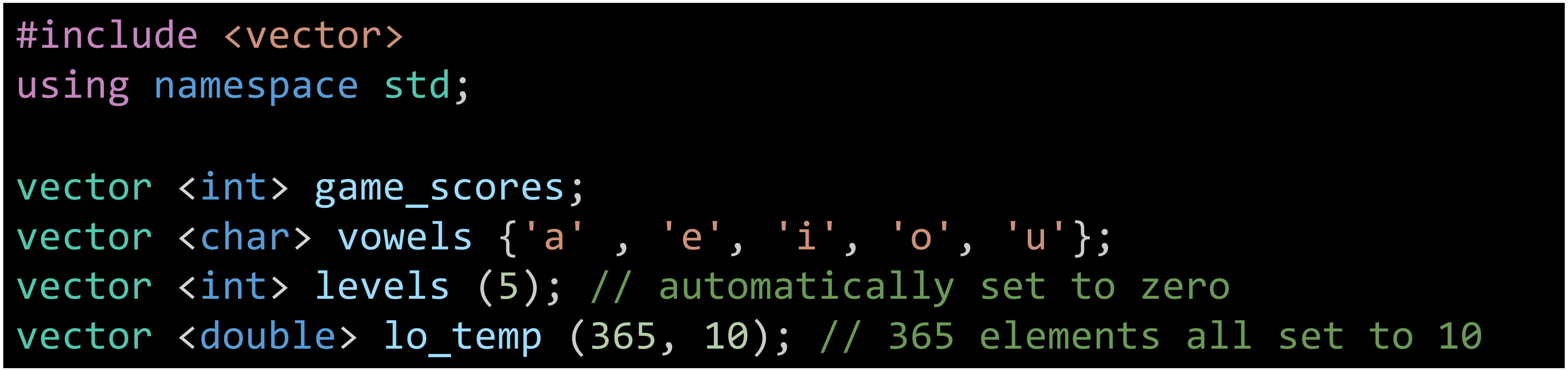
Vectors are objects



push_back	Add element at the end (public member function)
pop_back	Delete last element (public member function)

https://cplusplus.com/reference/vector/vector/

Vectors



```
#include <vector>
using namespace std;

vector <int> game_scores;
vector <char> vowels {'a' , 'e', 'i', 'o', 'u'};
vector <int> levels (5); // automatically set to zero
vector <double> lo_temp (365, 10); // 365 elements all set to 10
```

Vectors

- Dynamic size (*can grow or skink as needed at runtime*)
- Same type elements
- Stored in memory contiguously
- Can access individual element using their position
  - First element at index: 0
  - Last element at index size -1

Vectors

- Unlike array, elements are automatically initialized to zero unless stated otherwise
- If you use the subscript operator [], no bounds checking is performed
- Useful function that do bounds checking
- Very efficient

Vectors: accessing elements

- Accessing same as array with no bounds checking
  - vector_name [index]
    - score [1]

Vectors: accessing elements

- Accessing with bounds checking
  - vector_name**.at(index)**
    - score.at(4)
- Add element dynamically
  - vector_name**.push_back (element)**
    - score.push_back(100)

Demo Arrays and Vectors

>> DEMO