



# Welcome

---

## Manuel Cubuca

### Technical Trainer

[manuel@vistatrainingssolutions.co.uk](mailto:manuel@vistatrainingssolutions.co.uk)



# Agenda

---

- Introductions and register/attendance
- Health and Safety check
- Classroom structure and rules
- Apprenticeship overview
- Functional Skills information and support
- Training approach
- Core learning
- Main teaching – Lesson learning objectives



# Health & Safety

---

**If you feel unwell please let me know**

- Info about the building
- Careful with cables and drinks
- Get breaks away from the screen
- Make sure you are comfortable



# Classroom Structure & Rules

---

- Start at 10:00am
- Break in between (10-15 min)
- Lunch break at 12:30pm (for an hour)
- Break in between (10-15 min)
- We finish at 15:00pm
- Respect each other and work together
- Collaborate and create lasting friendships
- Follow the learner code of practice and the centre training rules



# Apprenticeship Overview

---

**Software Developer Assessment Plan**

**Software Developer Occupational Brief**

**Software Developer the Standards**

**Software Developer Knowledge Modules**



# Functional Skills Information & Support

---

- Apprenticeship Functional Skills English and Maths
- Classroom Support for English and Maths
- BKSB Website
- Functional Skills Courses on Talent LMS
- Embedding Functional Skills English and Maths

**BKSB Website**

**Talent LMS**



# Core Learning

---

*“Imagination is more important than knowledge  
but knowledge empowers imagination”*

**Class Discussion – Vista Blog**





# Learning Objectives

---

- Explain the role and function of the software development lifecycle (SDLC)
- Relate the seven generic stages of the software development lifecycle (SDLC)
- Show the main activities in each stage of the software development lifecycle
- Demonstrate the high-level deliverables from each stage of the software development lifecycle
- Describe the primary differences between the waterfall and agile software development methods
- Explain the respective strengths and weaknesses of each of the waterfall and agile software development methods
- Explain the respective strengths and weaknesses for using either the waterfall or agile software development method in a given case
- Relate the roles and responsibilities within software development and implementation
- Explain the structure of a software development team within an organisation
- Explain the team-working aspects that are needed to ensure effective delivery of projects





**Level 4 Software Developer Apprenticeship Programme**

# **Knowledge Module 1**

Technical Knowledge and Understanding



# KM1 objectives

---

## Pass the exam

3 modules:

- Software Development Life Cycle Concepts (50%) 3 Sections
- Software Development Methodologies (20%) 1 section
- Data Roles and Responsibilities (30%) 1 Section



# Level 4 BCS exam

---

- **Level 4 – equivalence to 1<sup>st</sup> year university level**
- **Knowledge level K3**
  - The candidate should be able to apply a topic in a practical setting and use the information in a new way. Typical questions would use: choose, demonstrate, employ, illustrate, interpret, operate, schedule, sketch, solve, use, write. (Source: [www.bcs.org/levels](http://www.bcs.org/levels))
- **40 question exam**
  - Multiple choice
  - 1 hour (15 minutes extra for non-native English speakers. Reasonable adjustments for disability allowed.)
  - Pass mark: 26/40 (65%)
  - No calculators allowed
  - Delivered online





**Level 4 Software Developer Apprenticeship Programme**

# **Software Development Lifecycle Design**

Technical Knowledge and Understanding



# Design (System vs UX/UI)

---

## System Design:

Once the requirements are understood, software architects and developers can begin to design the software. The design process uses established patterns for application architecture and software development.

Developers use proven **Design Patterns** to solve algorithmic problems in a consistent way. This phase may also include some rapid **Prototyping**, also known as a **spike**, to compare solutions to find the best fit. The output of this phase includes:



# Design (System vs UX/UI)

---

## UX Design:

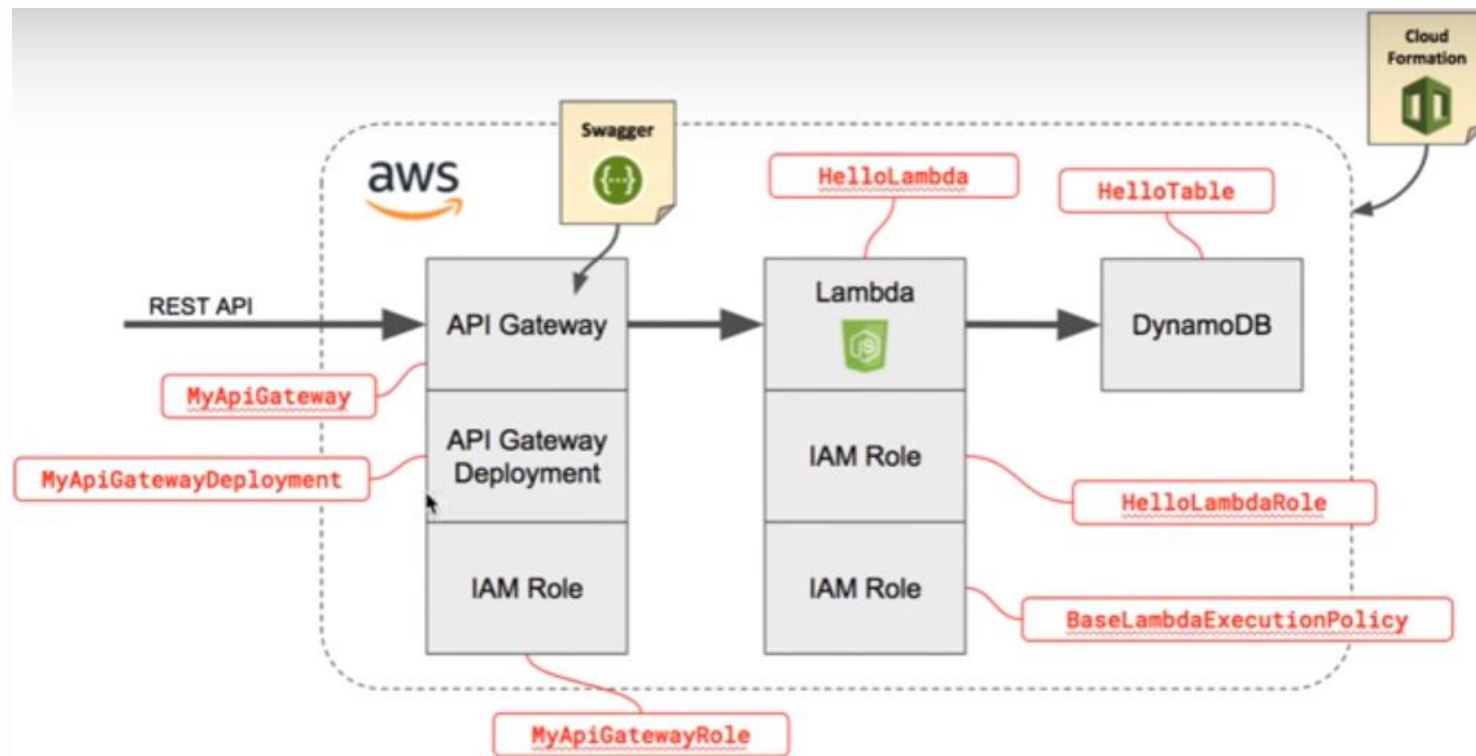
User experience design is the process of enhancing user satisfaction with a product by improving the usability, accessibility, and pleasure provided in the interaction with the product.

## UI Design:

User interface design or user interface engineering is the design of user interfaces for machines and software, such as computers, home appliances, mobile devices, and other electronic devices, with the focus on maximizing usability and the user experience.

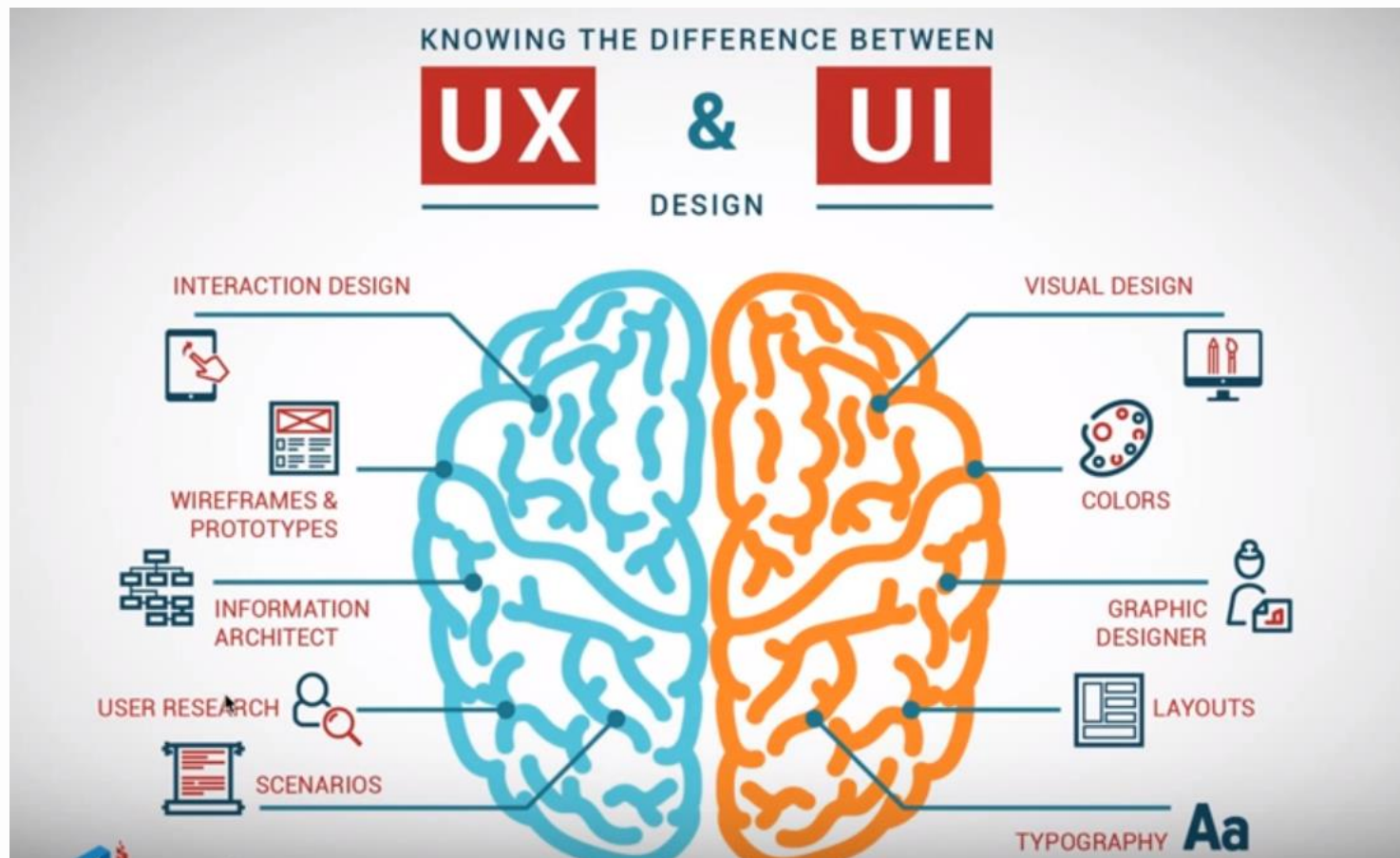


# Design (System vs UX/UI)





# Design (System vs UX/UI)



# Design (System vs UX/UI)

---

- **Design Patterns:**

General reusable solution to a commonly occurring problem

- Formalized best practices that the programmer can use to solve common problems when designing an application or system.
- Certain patterns in software design are known to recur
  - 23 fairly common ones
- Using software design patterns that are well-understood and have been successful in the past
  - Reduces design effort
  - Improves design quality
  - Helps developers communicate with each other more effectively
  - Can help reuse code



# Design (System vs UX/UI)

---

## Prototyping:

Prototyping is the quick creation of a mock-up of part of a system

- Aim is to help users give clear requirements
  - Human language is not particularly exact
  - Creating something that users can see and try to use may help them to be clearer about what they do and don't want

## Spike:

A Spike is a technical investigation to produce an answer in regards to some acceptance criteria on a PBI(Product Backlog Items) prioritised in upcoming Sprints. It's a great way to mitigate risks early and promotes fluid iterations later in the project. It allows the team ascertain feedback and develop an understanding on an upcoming PBI's complexity.





**Level 4 Software Developer Apprenticeship Programme**

# **Software Development Lifecycle Development/Coding**

Technical Knowledge and Understanding



# Development/Coding

---

This phase produces the software under development. Depending on the methodology, this phase may be conducted in time-boxed “sprints” (**Agile**) or may proceed as a single block of effort (**Waterfall**).

Regardless of the methodology, development teams should produce working software as quickly as possible. Business stakeholders should be engaged regularly, to ensure that their expectations are being met. The output of this phase is testable, functional software.



# Development/Coding

---

- Code is written according to the designs
- Developers unit test their own code and document the tests and results
- Bugs are fixed
- **Version control software** used to store code and manage changes
- Coding standards used to promote quality
- Programming tools used to facilitate development and increase developers' efficiency
- Peer reviews and retrospectives used to promote quality



# Code Development

---

- **Unit testing**
  - Individual programs tested in isolation from the rest of the system
  - Tests and results should be documented
  - Any bugs found should be fixed
  - Version control will be important to make sure that changes are properly managed
- **Automated testing** software should be used wherever possible
  - Quicker and more reliable (and less boring!) than manual testing
  - Especially useful for regression testing
    - Repeating previous tests after a bug fix or change to make sure nothing additional has been broken



# Coding Standards

---

- **What are coding standards?**
  - Ways of achieving 'best practices' in coding
  - Aim to improve quality of code and make testing and maintenance easier
  - Usually specific to an organisation
  - **May include:**
    - User interface style guides – consistent use of logos, fonts, colours, terminology for UI
    - Reporting standards - consistent use of logos, fonts, colours, terminology for reports
    - Code module design guidelines – may include
      - Design requirements for coupling, cohesion and complexity
      - Comments





# Programming tools

---

- **Debuggers**

- Programs that help to test and find and solve errors in other programs
- Often part of an IDE (Integrated Development Environment)

**Functions include:**

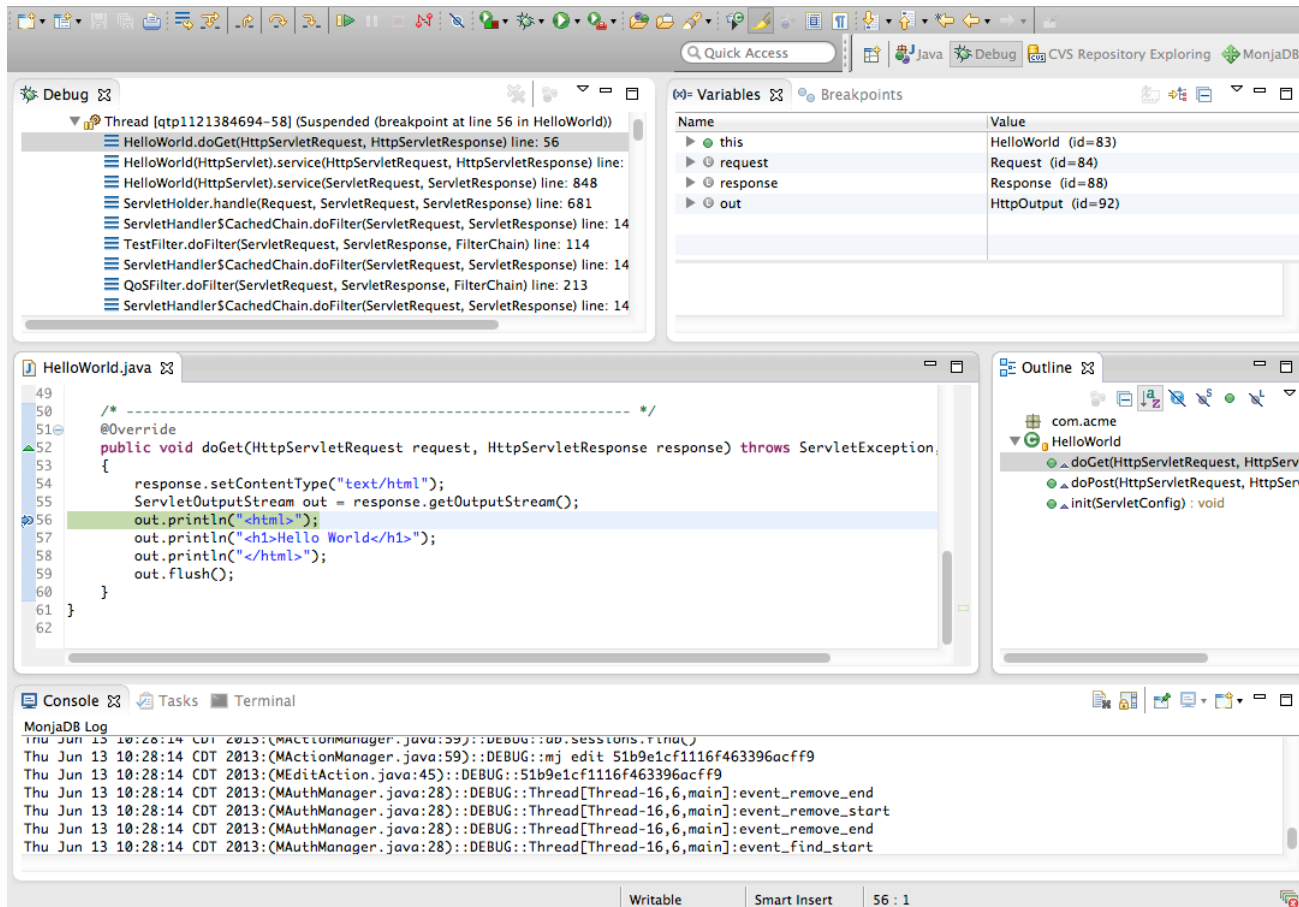
- Breaking – developer sets breakpoints - locations in code where execution will stop
- Single-stepping – going through code one line at a time
- Tracking values of variables

- **GUI designers**

- Software development tools that simplify the creation of GUIs by allowing the designer to arrange graphical control elements (often called widgets) using a drag-and-drop WYSIWYG editor
- Some produce usable code
- Save effort compared to specifying layout in code without visual feedback
- Can be used for prototyping



# Programming tools



The screenshot displays an IDE interface with the following components:

- Debug Console:** Shows the execution flow of a thread [qtp1121384694-58] (Suspended (breakpoint at line 56 in HelloWorld)). The stack trace includes:
  - HelloWorld.doGet(HttpServletRequest, HttpServletResponse) line: 56
  - HelloWorld(HttpServletRequest).service(HttpServletRequest, HttpServletResponse) line: 56
  - HelloWorld(HttpServletRequest).service(ServletRequest, ServletResponse) line: 848
  - ServletHolder.handle(Request, ServletRequest, ServletResponse) line: 681
  - ServletHandler\$CachedChain.doFilter(ServletRequest, ServletResponse) line: 14
  - TestFilter.doFilter(ServletRequest, ServletResponse, FilterChain) line: 114
  - ServletHandler\$CachedChain.doFilter(ServletRequest, ServletResponse) line: 14
  - QoSFilter.doFilter(ServletRequest, ServletResponse, FilterChain) line: 213
  - ServletHandler\$CachedChain.doFilter(ServletRequest, ServletResponse) line: 14
- Variables Panel:** Displays the current state of variables:
 

Name	Value
this	HelloWorld (id=83)
request	Request (id=84)
response	Response (id=88)
out	HttpOutput (id=92)
- Source Editor:** Shows the code for `HelloWorld.java`. The current line is 56, which is highlighted. The code is as follows:
 

```

49  /* ----- */
50
51  @Override
52  public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException,
53  {
54      response.setContentType("text/html");
55      ServletOutputStream out = response.getOutputStream();
56      out.println("<html>");
57      out.println("<h1>Hello World</h1>");
58      out.println("</html>");
59      out.flush();
60  }
61  }
62
      
```
- Outline:** Shows the class structure:
  - com.acme
    - HelloWorld
      - doGet(HttpServletRequest, HttpServletResponse)
      - doPost(HttpServletRequest, HttpServletResponse)
      - init(ServletConfig): void
- Console Log:** Displays the output of the application, including debug messages from the MonjaDB Log:
 

```

Thu Jun 13 10:28:14 CDT 2013:(MActionManager.java:59)::DEBUG::mj edit 51b9e1cf1116f463396acff9
Thu Jun 13 10:28:14 CDT 2013:(MEditAction.java:45)::DEBUG::51b9e1cf1116f463396acff9
Thu Jun 13 10:28:14 CDT 2013:(MAuthManager.java:28)::DEBUG::Thread[Thread-16,6,main]:event_remove_end
Thu Jun 13 10:28:14 CDT 2013:(MAuthManager.java:28)::DEBUG::Thread[Thread-16,6,main]:event_remove_start
Thu Jun 13 10:28:14 CDT 2013:(MAuthManager.java:28)::DEBUG::Thread[Thread-16,6,main]:event_remove_end
Thu Jun 13 10:28:14 CDT 2013:(MAuthManager.java:28)::DEBUG::Thread[Thread-16,6,main]:event_find_start
      
```



# Version Control

---

- Version control (aka source control or revision control)
  - Using a repository for code that helps manage changes in code
  - No code is perfect and it is inevitable that there will be changes in code for a system during development
  - **Exceptionally important to:**
    - Back code up
    - Manage multiple versions of code (allowing roll-backs to undo mistakes)
    - Record who made changes and when
    - Prevent two people from accidentally working simultaneously on the same program
    - Always be sure what the current version is
  - Helps programming teams work together effectively, especially where there are barriers to communication, e.g.
    - Large teams
    - Geographical separation



# Intro To Git

---

**Hands on Coding – Version Control**



# Part 1 Learning Objectives

---

- Learners would have a good understanding of:
  - What is HTML5
  - HTML5 new features and semantic elements
  - What are site maps, wireframes and style tile
  - Create webpage structure from scratch
  - Deploy HTML5 semantic tags
  - HTML5 attributes and their use
  - How to build the skeleton of a website



# Any Questions?

---

