MANCHESTER 1824

The University of Manchester

Software Engineering

In partnership with HyperionDev

BOOTCAMP OVERVIEW

Table of contents

Overview

If the idea of analysing a situation and seeing how it can be improved using software excites you, then software engineering may be the career for you. Software engineering involves more than just coding. This discipline uses principles applicable to a breadth of large-scale software systems. Ultimately, you'll be able to construct software solutions to solve specific business problems.

The engineering process involves working with stakeholders to understand the requirements and limitations of a software system. The software engineer analyses these requirements, and then designs, implements, deploys and maintains the software system.

No prior knowledge of coding is required when taking this bootcamp, as we help you progress from beginner to job-ready in only six months. Right from the start of the bootcamp, you're taught how to think like a programmer by developing systematic algorithms to solve various problems.

Going beyond software development

You'll learn how to write code that can interact with databases, and that uses established design patterns and algorithms to create useful software that solves real-world problems. Advanced-level outcomes also include being able to test, debug, deploy and maintain software systems, as well as guaranteeing their quality.

Throughout the bootcamp, you will be guided to develop the skills required to think beyond mere software development and deployment. You'll also learn to manage a software development project using agile development while communicating with technical and non-technical stakeholders. Here is where you learn how software forms part of a system's architecture, and how to apply best practice principles during the software development lifecycle.

The process

1. Log onto your personalised dashboard
2. Complete coding exercises online
3. Your code reviewer reviews your work within 48 hours
4. Perfect your coding over 3-6 months
5. Earn a certificate of completion
6. Begin your new career in tech

Outcomes of this bootcamp

- Design solutions to problems, and express them using pseudo-code and algorithms.
- Write useful code using Python and Java, two of the best programming languages to learn as they're used throughout the industry to create various web and mobile applications.
- Understand and apply computer science fundamentals, including data structures such as lists. Other fundamentals covered include algorithms for sorting and hashing, and using Big O Notation to analyse the performance of an algorithm.
- Use established algorithms to implement machine learning.
- Use agile development for software development projects.
- Design, plan, build, test, debug, refactor, deploy and maintain a software system.
- Use established design patterns and Git to ensure version control.
- Become job-ready in as little as 3 months.

Code reviewers in partnership with HyperionDev

Bootcamp code reviewers are expertly trained to integrate code review into the lives and bootcamp curriculum of participants. The on-demand code review method helps participants to become fluent in the language of their choice.

Our 1-on-1 code review approach works

Code review enables you to learn to code the right way through mastery of deeper aspects of software development that are a prerequisite for a career in coding. We help you master the deeper aspects of industry-level development and set the foundation for a lucrative career in coding.

Here's why learning through code review is smarter:

Don't make the same mistakes as computers

- Automated code checking is like spell check for computer programs. You can't write a world-class essay with just good spelling — you need the right tone, facts, grammar, and style. Only human-led code review can help you learn aspects of coding that are analogous to tone and style that will make you truly fluent as a developer — automated graders just can't help you learn this!

Get unstuck with on-demand technical help

- Our code reviewers will ensure you move at a steady pace by helping you debug your programs within 48 hours. This will help you to keep moving forward so that you stay on track.

Be exposed to the industry standards from day one

- Developers in the real world have their work assessed by a senior developer through the technique of code review. We're the only bootcamp in the world that exposes our bootcamp participants to this technique from day one so that you get an advantage in the job market.

We layer a proven, personalised approach to our code review

Industry experts tailored to your goals

- You'll work with experienced code reviewers who will guide you through 1-on-1 calls, career coaching, live chat, and email support.

Join a community of career-changers

- Learn as part of a cohort of participants all working towards ultimate career fulfilment. Join online group tutorials, community chats and meetups, and peer coaching.

Free of fear of failure

- Human-led code review builds trust with your educators and lets you progress at your own pace. Establish a safe space to discuss any roadblocks without fear of failure.

Why choose software engineering as a lucrative career?

Software engineering is a creative career that allows you to work with code and people, as well as hardware and other computer systems. This career places you at the heart of the digital economy, with endless scope for growth. Software engineers hold valuable skills that enable them to earn good salaries.

If you're looking for a career that is both rewarding and lucrative, software engineering delivers on both. However, those who develop software engineering skills can also choose to pursue other career paths, some of which we'll delve into next.

How we get you hired

We're with you every step of your journey, and our support doesn't end when you complete your bootcamp. Our career services are developed to help you stand out from the crowd, and grab the attention of top employers.

TECHNICAL CV AND PORTFOLIO

Receive technical assistance in getting your CV industry-ready according to acceptedbest-practice format.

INTERVIEW PREPARATION

Walk away with a new certificate as evidence of your skills and expertise in Software Engineering.

BOOTCAMP CERTIFICATE

Know what to expect when getting ready for that big interview with expert interview preparation from professionals who have been where you are.

JOIN OUR HIRING NETWORK

We work with select hiring partners and aim to help you land your first tech job interview after the completion of the bootcamp.

Potential career paths

When you enter the world of tech - and specifically the field of software engineering - the career options and role designations may seem intimidating at first. While the sky certainly is the limit once you learn to code, our bootcamps get you job ready for an entry-level role at a tech business. Below are some potential career path options you may want to consider working towards, or research further. Responsibilities include:

Back-End Developer

A Back-end developer builds, updates and maintains the server-side infrastructure, or "back end," of a website or application. They make sure that the assets on the user-end are functional and data is efficiently and securely stored.

Responsibilities include:

- Building and maintaining web applications.
- Assessing the efficiency and speed of current applications.
- Writing high-quality code.
- Managing hosting environments.
- QA testing.
- Troubleshooting and debugging.

A Back-end Developer in the United Kingdom can earn an average salary of £52,803 per year.

DevOps Engineer

A DevOps Engineer combines an understanding of both engineering and coding. They create and implement systems software to analyse data to improve existing systems, while working with various departments to create and develop new systems within a company. DevOps Engineers work to balance various aspects of a project, including complex issues such as programming and network building.

Responsibilities include:

- Implementing, maintaining, monitoring and supporting the IT infrastructure.
- Writing scripts for service quality analysis, monitoring and operation.
- Designing procedures for system troubleshooting and maintenance.
- Investigating and resolving technical issues by deploying updates/ fixes.
- Implementing automation tools and frameworks for automatic code deployment (CI/CD).
- Quality control and management of the code base.

A DevOps Engineer in the United Kingdom can earn an average salary of £47,698 per year.

Junior Software Engineer

Junior Software Developers are entry-level software developers that assist the development team with all aspects of software design and coding. Their primary role is to learn the codebase, attend design meetings, write basic code, fix bugs, and assist the Development Manager in all design-related tasks.

Responsibilities include:

- Assisting the development manager with all aspects of software design and coding.
- Attending and contributing to company development meetings.
- Learning the codebase and improving their coding skills.
- Writing and maintaining code.
- Working on minor bug fixes.
- Monitoring the technical performance of internal systems.
- Responding to requests from the development team.
- Gathering information from consumers about program functionality.
- Writing reports.
- Conducting development tests.

A Junior Software Engineer in the United Kingdom can earn an average salary of £29,540 per year.

Designer

Software Design Engineers are tasked with identifying software problems and designing programs to find solutions. They can either create a new product or iterations of existing software products to improve them. They gather data about the process and incorporate existing software solutions while determining and implementing the resultant software's parameters and limits.

Responsibilities include:

- Designs, codes, verifies, tests, documents, amends, and refactors complex programs/ scripts and integration software services.
- Takes responsibility for understanding client requirements, collecting data, delivering analysis and problem resolution.
- Designs software components and modules using appropriate modelling techniques following agreed software design standards, patterns, and methodology.
- Recommends designs which take into account target environment, performance security requirements, and existing systems.

A Software Designer in the United Kingdom can earn an average salary of £43,152 per year.

Systems Analyst

As a Systems Analyst, you'll use computers and related systems to design new IT solutions, modify, enhance or adapt existing systems and integrate new features or improvements in order to improve business efficiency and productivity.

Responsibilities include:

- Liaise closely with external or internal clients.
- Analyse clients' existing IT systems and business models.
- Map and document interfaces between legacy and new systems.
- Understand software development lifecycles.
- Translate client requirements into highly specified project briefs.
- Identify options for potential solutions and assess them for both technical and business suitability.
- Conduct requirements analysis and prepare specific proposals for modified or replacement systems.
- Develop solutions and related products.

A System Analyst in the United Kingdom can earn an average salary of £37,326 per year.

Software Tester

As a Software Tester, you'll be involved in the quality assurance stage of software development and deployment. You'll conduct automated and manual tests to ensure the software created by developers is fit for purpose and any bugs or issues are removed from a product before it gets deployed to everyday users.

Responsibilities include:

- Work with software developers and project support teams.
- Identify business requirements.
- Plan projects.
- Monitor applications and software systems.
- Carry out stress testing, performance testing, functional testing and scalability testing.
- Write and execute test scripts.
- Run manual and automated tests.
- Test in different environments including web and mobile.
- Write bug reports.
- Assess code.

A Software Tester in the United Kingdom can earn an average salary of £30,735 per year.

Structure of the bootcamp

This bootcamp helps you progress from learning the basics of programming and acts as path to a rewarding career in software engineering. Proceed from novice to job-ready, and land the successful role you deserve.

Bootcamp prep (Before you start)

- Learn about the software development sector and how we support you in achieving your development goals. Start programming with Python to attain a clearer idea of whether a career in the software development industry is really for you.

Introduction to Programming (Beginner level)

- Get to grips with the fundamentals of programming and the Python programming language. You also learn the basic concepts and master fundamental skills needed to code in Python.

Introduction to Software Engineering (Intermediate level)

- Understand how industry professionals develop software by exploring the best practices they use. You're also challenged to work on your programming skills, enabling you to deliver the most effective solutions for clients.

Advanced Software Engineering (Advanced level)

- Take on the more advanced software engineering concepts and explore aspects such as deployment and maintenance best practice, quality assurance, Big O Notation, machine learning, and algorithms.

Career Readiness and Employability (Post Bootcamp completion)

- Once you have completed your bootcamp, we provide career support and guidance, including interview preparation and CV review, to equip you with technical skills and professional career development tools to succeed in your job search.
- We introduce the participants who have completed the bootcamp to the industry through various networking events, career expos and job opportunities with our hiring partners. Most of our participants get hired within six months of completing the bootcamp with our support and mentorship.

Breakdown of syllabus

The bootcamp is structured to allow you to start coding as soon as possible.
Tasks are designed to:

- Teach you the theory needed to develop your skills.
- Give you the platform to practise implementing your new knowledge by completing one or more practical activities.

Remember, you're never alone. You can contact one of our expert code reviewers for 1:1 support whenever you need help with a task. The code you submit for each task is reviewed by a code reviewer who is an industry expert, to help improve efficiency and quality of code.

Introduction to programming

Tasks: 30 **Capstone projects: 4**

Tasks no.	Task name	Description
1	Thinking Like a Programmer - Pseudo Code I	Learn how pseudo code can help you clarify your thoughts and properly plan your programs before writing any code.
2	Thinking Like a Programmer - Pseudo Code II	Delve further into algorithm design and representation.
3	Your First Computer Program	Get acquainted with Python, the powerful, easy to learn and extremely popular, high-level programming language.
4	Variables - Storing Data In Programs	Learn how to store and interact with the data in our programs using variables.
5	The String Data Type	Learn how to store and manipulate text using the string data type.

Tasks no.	Task name	Description
6	Numerical Data Types	Explore the different types of numbers used in the Python programming language.
7	If Statements and the Boolean Data Type	Learn how to use the if statement to make decisions in your program.
8	Beginner control structure - Else statements.	Learn how to control the order in which statements are executed using the else statement.
9	Beginner Control Structures - Elif statements.	Learn how to check for multiple conditions using elif statements.
10	Logical programming - Operators.	Learn how to tell the compiler how to perform specific mathematical, relational or logical operations using operators.
11	Capstone Project I - Variables and Control Structures.	Put your knowledge of variables and control structures to the test by creating an investment calculator.
12	Beginner Control Structures: While loop	Learn how to execute a block of code repeatedly until a given condition returns false using while loops.
13	Beginner Control Structures - For Loop	Learn how to use the for loop to repeat a section of code a specified number of times.
14	Towards Defensive Programming - Error Handling	Discover the different types of errors that might occur in your programs and how to handle them.
15	String Handling	Learn how to manipulate text using Python's built-in functions.
16	Beginner Data Structures - The List	Discover the most frequently used and versatile collection data type used in Python - the list.

Tasks no.	Task name	Description
17	Working With External Data Sources - Input	Create smarter programs by learning how to read data from text files.
18	Working with External Data Sources - Output	Learn how to write data to text files.
19	Capstone Project II - Files	Put everything you've learnt about files to the test in this comprehensive task.
20	Beginner Data Structures - Lists and Dictionaries	Learn how to manipulate lists and become acquainted with dictionaries.
21	Beginner Programming With Functions - Using Built-In Functions	Learn how to use Python's built-in functions to provide better modularity for your programs and encourage code reuse.
22	Beginner Programming with Functions - Defining Your Own Functions	Create your own Python functions to carry out specific tasks.
23	Hypothesis-Driven Debugging With the Stack Trace	Learn to debug methodically and move away from trying to resolve errors randomly.
24	Capstone Project III - Lists, Functions and String Handling	Use all the knowledge you have gained so far throughout this bootcamp to create a useful program.
25	Introduction to Python - Data Structures - 2D Lists	Discover the most frequently used and versatile collection data type used in Python - the list.
26	Applied Recursion	Explore the concepts of recursive programming and how to "think recursively".
27	Towards Defensive Programming II	Learn how to guard against errors you don't expect.

Tasks no.	Task name	Description
28	Introduction to OOP I - Classes	Introduction to the principles of Object Oriented Programming (OOP).
29	Introduction to OOP II - Inheritance	Learn how you can improve the modularity and reuse of code using inheritance, and the critical role it plays in Python's object system.
30	Capstone Project IV - OOP	Apply the fundamentals of object-orientation to solve a simple problem.

Introduction to Software Engineering

Tasks: 23 **Capstone projects: 2**

Tasks no.	Task name	Description
1	The Software Process	Delve into the concepts of the software development process and the software process models.
2	Agile Development	Learn about agile development and one of the most popular agile methodologies - Extreme Programming.
3	System Requirements and Design	Explore best practice guidelines for defining your product and UX/UI design

Tasks no.	Task name	Description
4	System Architecture	Discover the various components that make up and interact with a software system.
5	Quality Management	Discover how to ensure that your developed software is both dependable and secure.
6	Deployment and Maintenance Best Practice	Discover the best practice guidelines for ensuring effective deployment and maintenance of software systems.
7	Introduction to Network Protocols and System Architecture: HTTP and Client-Server	Learn how computers communicate with each other over the internet using the HTTP protocol, and learn the commonly used client-server architecture for transferring information using HTTP.
8	Working From The Command Line	Learn to use the command line for web development, including basic commands and functionality used with the command line.
9	Introduction to Databases	Compare relational, graph, and NoSQL databases.
10	Design and Build Relational Database	Design a database by applying normalisation principles. Create relational databases.
11	Working With SQL	Learn about agile development and one of the most popular agile methodologies - Extreme Programming.
12	SQLite	Get comfortable with SQLite, a self-contained, public domain SQL database engine.

Tasks no.	Task name	Description
13	Capstone Project I: Databases	Learn how to communicate with your database using SQL and MySQL.
14	Introduction to Web Development	Learn what the web is. To write programs that run on the web, we first need to grasp what it is, and how people interact with it.
15	HTML Overview	Learn to use HTML to add content to a webpage.
16	CSS Overview	Use CSS to improve the appearance of your webpage.
17	Bootstrap: Build Attractive Pages Faster Using Bootstrap	Learn how to style like Twitter does.
18	Django I	Introduction to Django.
19	Django II	Build a blogging application.
20	Django III	Build a poll application.
21	Django IV	Extend the poll application.
22	Django V	Authentication and Authorisation.
23	Capstone Project II: Django	Build a Django web application.

Data Science, Algorithms and Advanced Software Engineering

Tasks: 13 **Capstone**

Tasks no.	Task name	Description
1	Sources of Data	Learn how to extract and import data from different sources (JSON, XML, CSV).
2	Sorting and Searching	Learn about data sources and types, data structures, and how to use these, including ordering and finding data in different types of data structures.
3	Version Control I: Introduction to Version Control and Git	Explore the Git version control system and the GitHub collaboration platform.
4	Version Control II: Git Basics	Dive into using Git and discover how to set up a repository, use common Git commands, commit a modified file, view your project's history, and branch.
5	Version Control III: Deployment of Static Websites	Learn how to deploy your websites using GitHub Pages.
6	Build Your Brand I	Use GitHub to start building a portfolio of work that you can share with others to showcase your skills.
7	Version Control IV: Pipelines	Learn about how Git is used in real-world collaborative project.

Tasks no.	Task name	Description
8	Containers: Docker	Learn Docker, a service that is used to build and share applications regardless of the platform it is run on.
9	Software Documentation	Learn how to store and interact with the data in our programs using variables.
10	Capstone I	Add version control to your Django project, document the project, and containerise it.
11	Introduction to NLP	Get acquainted with Natural Language Processing by learning about parts of speech, parsing, and how to install and start using spaCy.
12	Semantic Similarity (NLP)	Learn about semantic Similarity, a popular application of NLP widely used for social media analysis.
13	Capstone Project II: NLP	Utilise your newly acquired knowledge of semantic similarity and natural language processing in this final capstone project.