Software Development Syllabus

InfinityLabs R&D

Infinity's Curriculum Philosophy

Our goal is to provide all participants the skills and know-how to be ever-evolving Software Development experts. Even as they learn one particular technical stack, Infinity provides participants with what it takes to be well-rounded, "can-do" professionals who can independently migrate to new platforms and technologies throughout their careers. Our training equips them with the fundamentals and resources that enable this flexible technical agility.

Our goal and distinguishing hallmark is the transformation of our trainees into consummate professionals, who are not only capable of independently learning new technologies, but whose work habits and professionalism reflect the highest level of excellence. Our proprietary syllabi are designed to enable our participants to reach those objectives.

Open Lab Syllabus – Fundamentals of System Software Development

Our software development training tracks begin with our Open Lab (OL) syllabus that encompasses the core fundamentals and pre-requisites that every software developer needs to know, including:

- ✓ Essential concepts and behind-the-scenes workings of system programming
- ✓ Tools and resources
- ✓ Procedural programming languages
- ✓ Agile software development methodologies
- ✓ Professional skills (effort optimization, accurate time estimation)
- ✓ Software development traps & pitfalls
- ✓ Soft skills (personal effectiveness, inter-personal skills, communication)
- ✓ Technology research skills
- ✓ Software development as an individual and as part of a team

Compatible even for graduates of Computer Science programs, our syllabus is relevant for today's high-tech industry and is far beyond the scope of what one studies in academia. It provides a solid foundation for branching out in a multitude of different software development domains.

Goals and high-level skill set:

- Abstract data types
- NASA coding standard
- Revision control, Git & GitLab
- Linux, Linux shell & Bash
- Bitwise operations
- C programming language
- Structured programming
- Algorithm theory and real-world implementation
- API development
- Accurate time estimation
- Interface design
- Data structure theory and characteristics vs real-world implementation
- Calculating orders of complexity in theory and in practice
- Shared objects / DLLs
- Adaptive vs. predictive methodologies
- Industry-quality deliverables
- Unit testing, regression testing, smoke testing
- Agile software development & essential practices for Agile projects
- Asymptotic analysis
- Debugging tools & release mode development
- Software Development Lifecycle (SDLC)
- Reading code & how to review code

- SW engineering processes and methodologies
- Build process internals
- Operating system theory & real-world system programming
- Multi-threaded and multi-process systems
- Standard libraries
- DoD vs. AC (definition of done vs. acceptance criteria)
- Spiral development
- Low-level programming
- Greedy, sliding window & backtracking algorithms
- Crash dump analysis
- Code reusability
- Word boundaries and structure padding
- Remote debugging
- Planning with pseudo-code
- CI / CD principles and tools

2nd Stage Training Tracks

After completion of the Open Lab, participants branch out into different learning tracks – each of which provides basically the same fundamentals (common to all tracks). Our philosophy is that the selection of who goes to which track is not a career or employment decision, but an educational strategy that considers the best way for each individual trainee to proceed and learn the next set of fundamentals.

Goals and high-level skill sets common to all tracks:

- Advanced system programming
- Object Oriented Programming (OOP)
- Cross-platform development
- Distributed systems
- Networking & network software development
- Design patterns
- Building user-friendly interfaces
- Multi-tier architecture
- Multi-paradigm development
- Debugging complex systems
- Project architecture & design

- REST / RESTful API
- UML
- Testing strategies
- Asynchronous communication
- Client / server
- Cloud architecture & cloud storage principles
- Event-driven development
- Code refactoring
- S.O.L.I.D. principles
- Master – Minion topologies
- Strictly layered systems / layered architecture
- Real-world software engineering practices
- Separation of Concerns (SoC)
- Backward compatibility
- Quality-driven development
- Dynamic programming
- Technical presentation

R&D Syllabus – Multi-Disciplinary Development

Our R&D (RD) syllabus provides a 'bottom-up' approach to learning – with in-depth theory and an array of under-the-hood technologies and respective skills that are relevant for a wide range of software development disciplines. Topics covered include systems, networks, architectures & topologies, environments, frameworks, professional tools, coding styles, development environments, the software development lifecycle (SDLC), additional programming languages and implementation techniques.

Goals and high-level skill set:

- Drivers & kernel
- Framework development
- Complex coding scenarios
- Multi-platform systems
- Concurrency
- Real-world software engineering practices
- Virtualization & separating a service from its physical delivery

- Master-Minion topology
- C++ internals
- RAID standard
- Defensive coding
- Internals and dependencies
- Server-side / Web server
- C++ programming language
- IPC
- Functional decomposition
- File system
- Internals and dependencies
- Multiple autonomous components
- Block device
- MVC / MVVM
- Embedded targets
- Conventions & casts
- Ubiquitous computing
- System Call Interface (SCI)
- STL / Boost
- IoT utilization
- Rule-based programming
- Optimizers / profilers
- Exceptions
- Scope lock
- Event loops
- Template metaprogramming (TMP)

Full Stack – Back-End (FS) Syllabus

Our Full Stack – Back-end syllabus provides a ‘top-down’ approach to learning, with a comprehensive exposure to the technologies, tools and trends of back-end Web development and Web services, including specialized and best programming practices, multiple types of system development, databases, APIs, languages, UI / UX basics, networks, and cloud computing.

Goals and high-level skill set:

- Backend development
- Java
- Garbage collector
- Static and dynamic binding
- Chain-of-responsibility
- State machines
- Abstract classes
- Error handling & exceptions
- Collections
- JVM
- Shared objects / JARs
- Interfaces
- Inner classes
- Generics
- Future objects
- Built-in data structures
- Runnable & callable
- Concurrency
- I/O system
- Java NIO
- Gson / JSONf
- Maven
- Message broker & queue
- JavaScript
- JDBC
- JAR file
- Rule-based programming
- Complex coding scenarios
- CRUD
- Functional programming
- Databases design
- SQL
- NoSQL
- Ubiquitous computing
- XML
- Reactive programming
- Apache / Tomcat

- WebSocket
- Fault tolerance
- Web server

Full Stack – Front-End (FE) Syllabus

Our Full Stack – Front-end syllabus provides a 'top-down with visual aides' approach to learning. It covers all areas of the user interface (UI), including relevant programming languages and tools, user-centered design (UCD) and user-experience (UX) concepts, design principles, intuitive layouts and presentation, and best practices for implementation.

Goals and high-level skill set:

- UI design tools and techniques
- Data-driven design
- Client build tools
- UI / GUI & materials libraries
- HTTP / HTTPS
- Adaptive design
- JavaScript libraries and frameworks
- ECMAScript 6
- Multimedia
- DOM
- Browsers
- Cypress
- Gestures
- Layouts
- Express
- AJAX
- E2E testing
- Input events
- Animation
- Graphics
- Immutability
- Jest
- User Experience (UX)

- Unidirectional data flow
- Input controls
- CSS
- HTML
- SQL vs NoSQL
- Sass
- State management
- MongoDB
- User-centered design
- Webpack
- npm
- React
- Redux
- Regex
- Responsive design
- Built-in data structures
- Router
- BaaS
- Event loops

Specializations

After completing the training, we assist our graduates to secure their first positions, and if a position should require some extra training to get up-to-speed with a specialized technology, Infinity offers a dedicated technical completion – a period of 1—5 weeks of additional training in the required area, to ensure a smooth transition for our graduate on their first job.

Specializations include (but not limited to):

- Visual Studio
- Emacs
- AWS / OpenStack
- Mobile technologies
- Python
- RabbitMQ
- Memory ordering

- I/O Ports
- Memory management & MRC
- vs. ARC
- MySQL
- Kernel compilation
- Xcode
- NodeJS
- Data science
- PHP
- SQLite
- C#
- Spring
- VxWorks
- LINQ
- Android
- iOS
- Hibernate
- Drivers
- SOAP
- Ruby
- Windows
- Swift
- Software / hardware
- fragmentations
- Bluetooth
- Code blocks
- AngularJS
- Multimedia
- Energy diagnostics
- Network analysis
- .NET
- GitHub
- Facebook / Google authentication