COURSE FROM IT SCHOOL №1 ON EMPLOYMENT RATES

«MANUAL TESTING FROM SCRATCH + AI-APPLICATIONS TESTING + | INTERNSHIP IN 🇺🇸 IT-COMPANY»

DURATION OF TRAINING: 4 MONTHS + 4 MONTHS OF INTERNSHIP

You deserve the best

I. COURSE SYLLABUS

Entrance examination (vocational guidance course)

why it is needed: to answer the student's internal question "is this the right profession for me or not?", to test the student's "tester mindset", to assess the student's ability to cope with complex learning experiences.
result: assurance that the student will not waste time/money, mentors will not waste resources, and the education will be successful with a 90% probability of success

Stage №1	Interactive textbook, online tests, and homework assignments to understand the QA profession with live feedback
Stage №2	Screening for a QA mindset - solving IT problems in a team Zoom call, just like in a real IT project
Stage №3	One-on-one interview on IT prospects with a lead mentor with 20+ years of QA experience

training module

Relational databases for QA [DB]

why it's needed: to find where the application stores data and use that data for testing; to understand how the application writes and reads data to speak the same language with the developer (no special magic there)

student knows: what DB are for, how their structure is organized, datatypes, ER, what SQL is, what are the database manipulation commands

has the skills to: analyze a DB schema, retrieve data by JOIN tables and using COUNT, SUM, HAVING, GROUP BY operators and NOT, LIKE, BETWEEN conditions

training block

Relational databases (DB1)

Theory (textbook 02) + quiz	**All examples should be illustrated using the FTB DB scheme.** 1. Where to store information: Excel sheets, files, paper catalogs; Data storages in software (databases relational and non-relational; why relational DB are popular); QA tasks of working with relational DB. 2. What DB consists of: What are tables; Columns and fields; Objects/properties/datatypes (on an example of passenger and flight, other possible datatypes (char, nchar, nvarchar, int, bigint, float, decimal, datetime, date, time, Boolean)); Unique identificator (ID, its int type, autofill); Fields constraints (datatype, field length, mandatory and not mandatory fields). 3. Relationships between tables: What is a relationship (on an example of a relationship between passenger and flight; How such a relationship could be realized in DB); Why do we need relationships (DB normalization); What relationships between tables could be (1 : 1, 1 : many, many : many); Public and foreign keys; How many:many relationships should be realized (auxiliary table with IDs). 4. ER diagrams and DB tools: ER-diagram definition; Tools for working with DB and ER diagrams; DBeaver (how to install it; how to connect to DB - each student uses their credentials; how to generate ER diagram in DBeaver; students should look at the FTB ER diagram in DBeaver); Students should observe all the elements on the FTB ER diagram (relationships, PK, FK, datatypes on ER-diagram; Getting information transitively; cases, when 2 tables are connected several times by different foreign keys; not connected tables).
Practical workshop	15 students solve real IT tasks, while a lead mentor with 10+ years of IT experience guides and assists them: - Read FTB ER-diagram, match with objects on UI - Update FTB ER-diagram for hypothetical new functionality
Homework	- Draw an ER diagram of an online library

training block

SQL queries (DB2)

Theory (textbook 03) + quiz	**All examples should be illustrated using the FTB DB scheme.** 1. What is SQL and why do we need it: What could be done with SQL; Typical testing SQL tasks of working with the information, stored in a single table (get, add, change or delete; examples from passengers and flights). 2. Getting information from a single table: Getting all the table data (SELECT * from table); Filtering information (WHERE, NOT, LIKE and wildcards, BETWEEN, =, <>, !=, DISTINCT, IS NULL/IS NOT NULL); Sorting information (ORDER BY DESC, ASC, combinations); Combinations of conditions and sorting. 3. Data manipulation operations with a single table: Adding information (INSERT); Changing information (UPDATE); Removing information (DELETE); Why not use DELETE FROM without conditions; DB permissions (Read-only, edit permission, delete permission, why juniors often have read-only permission). 4. QA Engineer testing tasks: How to test the correct implementation of various relationships in an application (1:1, 1:many, many:many).
Practical workshop	15 students solve real IT tasks, while a lead mentor with 10+ years of IT experience guides and assists them: - SELECT - WHERE - INSERT, UPDATE, и DELETE
Homework	- SELECT without JOINS - WHERE - INSERT

training block

SQL Functions and JOINs (DB3)

Theory (textbook 04) + quiz	**All examples should be illustrated using the FTB DB scheme.** 1. SQL Functions: Why do we need SQL Functions; Aggregate functions (SUM(), AVG(), COUNT(), MIN(), MAX()); Date functions (NOW(), CURDATE(), CURTIME(), DATEDIFF()); String functions (CONCAT(), CONCAT_WS(), LOWER(), UPPER(), SUBSTRING(), TRIM()). 2. JOINS: Why do we need JOINS: A brief reminder about the DB relationships; What is JOIN; Connect 2 tables by FK without conditions with INNER JOIN; LEFT JOIN; RIGHT JOIN; COALESCE(); Aliases and why do we need them. 3. Grouping information by a field: Why do we need it; GROUP BY and HAVING (using GROUP BY with and without HAVING, difference between HAVING and WHERE).
Practical workshop	15 students solve real IT tasks, while a lead mentor with 10+ years of IT experience guides and assists them: - SQL Functions - JOIN
Homework	- Write a query that will show the flight number, arrival airport name, and departure airport name values

training module

LINUX FOR QA [L]

why it's needed: to find out what the application on the server was saying before the crash and to be able to run the application in a test environment for testing
student knows: Linux: advantages as a server system, **FS:** basics (inc. absolute/relative paths)
has the skills to: log in to the server, **env:** determine server name, OS type & free disk space, **FS:** directories navigation, **log:** deep analyze, **CLI:** run app with parameters

training block

Linux Commands and Filesystem (L1)

Theory (textbook 06) + quiz	1. What is Linux in QA Life: Servers; CLI. 2. What are Linux commands and their structure: id, uname, cat comm. structure (on egrep example). 3. Linux filesystem and objects: FS structure; FHS; absolute/relative paths; pwd, cd, ls, cp.
Practical workshop	15 students solve real IT tasks, while a lead mentor with 10+ years of IT experience guides and assists them: cd, pwd, ls, man
Homework	cd, pwd, ls, man

training block

Text Processing in Linux (L2)

Theory (textbook 07) + quiz	1. Linux file types: File types; Text files (\r\n). 2. Text processing in Linux: cat, less, egrep, head, tail, wc, sort, uniq, echo. 3. Output redirections: stdout/stderr; Merging; /dev/null. 4. Pipes and stdin: Use case; Stdin; Important notes.
Practical workshop	15 students solve real IT tasks, while a lead mentor with 10+ years of IT experience guides and assists them: Input-output redirection (>, 2>, 2>&1), cat, less, egrep, head, tail, wc, sort
Homework	Output redirection, cat, egrep, head, tail, wc, sort, pipes

training module

NETWORKS FOR QA [N]

why it's needed: to detect simple (and most common) problems with an application not working due to local & global network issues

student knows: basics of client-server model, IP addresses and ports, basic routing, DNS, MAC addresses, localhost

has the skills to: detect problems for web-server apps with unavailable IP address (ICMP ping), closed port of the service or DB, DNS problems, using cURL

training block

Networks overview (N1)

Theory (textbook 11) + quiz	1. Networks and networking tasks in QA: Preparing the environment; The “Client-server” model. 2. Basic network concepts: Addresses and routing; IP addresses; Routing; DNS (ipconfig / ip / ifconfig, route / netstat, nslookup / dig / host, tracert / traceroute). 3. Hardware: MAC address; Network adapters and IP addresses (arp, ipconfig / ifconfig / ip).
Practical workshop	15 students solve real IT tasks, while a lead mentor with 10+ years of IT experience guides and assists them: - Network usage in QA tasks
Homework	windows: ipconfig, route, tracert, nslookup linux: ip, host/dig, curl, traceroute, host/dig

training block

Network Layers and IP addresses (N2)

Theory (textbook 12) + quiz	1. Introduction to TCP/IP: Layers, TCP/IP stack in a nutshell, Encapsulation, cURL. 2. IP: IP Addressing, Routing, Where are IP addresses coming from, NAT, Localhost (for Docker). 3. DNS: Domain Name System (ipconfig / ip / ifconfig, route / netstat, nslookup / dig / host, tracert / traceroute).

Practical workshop	15 students solve real IT tasks, while a lead mentor with 10+ years of IT experience guides and assists them: - IP addresses and netmasks: how they may be configured - Routing: how the packets are sent, and look under the hood via very basic tools - DNS: how you can operate with that effectively
Homework	host, traceroute, ip, ipconfig, route, dig

training module

SOFTWARE ARCHITECTURE FOR QA [Tr]

why it's needed: to realize that many defects are highly dependent on the architecture of the application
student knows: standalone, 2-tier, 3-tier architecture; algoritm of bug localization; ready to use this knowledge on the following modules R,W
has the skills to: analyze the application design, log analysis for high-level level cause determination; determine the number of tiers based on log analysys

training block

Software Architecture (Tr1)

Theory (textbook 19) + quiz	1. 2-Tier Application Architecture: The principles of operation, Possible issues (Operating system, Network, Client-side troubles). 2. 3-Tier & N-Tier Application Architecture: The principles of operation, Possible issues (Network: Upper tier to Data tier, Data tier); N-Tier Applications; The principles of operation; Load balancers. 3. Application logs for localizing server defects: Logging levels; Examples of using logs to localize problems.
Practical workshop	15 students solve real IT tasks, while a lead mentor with 10+ years of IT experience guides and assists them: - What does the standalone server application look like? - How do the components of a 2-tier server application communicate with each other and clients? - Which additional complexity does the 3-tier architecture introduce?
Homework	- Log analysis to delect high-level root determination and number of tier levels

training module

REST & API FOR QA [R]

why it's needed: to detect a defect not when it has already appeared on the application UI (it is very expensive for development), but much earlier
student knows: API, Open API, URL, URN, URI, REST principles, JSON, data exchange rules, server response codes
has the skills to: send valid/invalid requests(similar to DB queries) to the server-under-test, read JSON responses and write JSON requests, validate responses; use cURL (on advanced level), Swagger, Postman (manual), Open API ???

training block

HTTP, REST, API (R1)

Theory (textbook 15) + quiz	1. HTTP Protocol: Protocol (definition and usage); Hypertext and Hyperlink (definitions and difference); HTTP and HTTPS (definitions, usage, difference, default ports, encrypting, certificates SSL/TLS and keys, CSR, CA); URL, URN, URI (definitions and difference, structure of each); HTTP Methods (GET, POST, PUT, PATCH, DELETE, HEAD, OPTIONS, CONNECT, TRACE, structure of HTTP requests and responses, naming: CRUD, exercises with CURL for each method, response codes, headers, localhost, body). 2. Tools: cURL (advantages and limitations); OpenAPI (definition and documentation); Swagger (usage for testing of public API https://ftb.mentorpiece.org/swagger-ui/index.html); Browser plugins (Boomerang, RESTED). 3. JSON: Definition; Examples; Data types (string, number, Boolean, object, null, array). 4. API: Definition, Types of API (hardware, OS, Web - SOAP, REST, GraphQL, gRPC); API protocols, API hosting options (server, clouds); How API works (examples); API documentation structure (description, endpoints, methods, parameters, request/response format, exceptions, examples, limits and quotas); API documentation types (static and dynamic); Examples of public APIs (https://ftb.mentorpiece.org/swagger-ui/index.html); Authorization (who you are) and authentication (what you can do) of API (definitions, difference, usage, tokens, API keys).

Practical workshop	15 students solve real IT tasks, while a lead mentor with 10+ years of IT experience guides and assists them: - Practice in Data transferring - curl GET requests. Requests with headers. - Practice in API - Practice with JSON - read/create json objects - Practice with API - curl GET requests to public APIs, Authentication using crul
Homework	Basic: - Create a valid JSON object from the verbal description - Common manual actions with JSON objects for QA Advanced: - Use cURL or just a browser to interact with the weather API - Get current weather data for selected location - Submit the JSON for grading

training block

HTTP, REST, API Testing Tools (R2)

Theory (textbook 16) + quiz	1. REST: REST vs. RESTful (definitions and difference); Resources, Actions, URLs, Data formats, REST methods and their connection with HTTP methods (GET, POST, PUT, PATCH, DELETE); REST request structure (Verb - Method, Payload - Header and Body, HTTP version, URI, payload in different methods); REST response structure (response code, HTTP version, response header and body); examples. 2. Postman: Manual testing with Postman (web version, desktop version, usage, collections usage, variables, testing with Postman on an example of public API); Postman automation (scripts on Java Script, snippets). 3. Testing of REST API: Typical situations (goal of REST API testing, typical errors, typical checks, check of supported methods); POST/PUT testing (request and response structure, mandatory and optional parameters, supported data types, limits and boundaries, response codes, examples); GET testing (request and response structure, mandatory and optional parameters, supported data types, limits and boundaries, response codes, examples); DELETE testing (request and response structure, mandatory and optional parameters, supported data types, limits and boundaries, response codes, examples); Positive scenario (several methods testing together).

Practical workshop	15 students solve real IT tasks, while a lead mentor with 10+ years of IT experience guides and assists them: - Swagger and Redoc: making a request, authentication, authorizationusing web UI - Postman, Insomnia: make a request, import/export/create collections
Homework	- Testing pet store user resource using Postman (import collections) - Alaska Bears API: GET, POST, PUT, DELETE

training module

WEB UI FOR QA [W]

why it's needed: to detect a defect on the UI
student knows: HTML, JS, CSS
has the skills to: can use DevTools & Proxy servers for GUI visual & functional testing

training block

WebUI (W1)

Theory (textbook 17) + quiz	All examples should be illustrated with the test site www.example.com 1. What is UI: UI types (GUI, CLI, Menu Driven, Form based, Touch, Voice, WebUI as a special case of GUI); How the web page works. 2. Components of a web page - HTML, CSS, JS: Common structure of the web page (HTML, CSS, JS - common schema); What is HTML; Example of HTML code; How to open HTML code of www.example.com; The explanation of each html tag on www.example.com; Other basic tags (ul/li, img); AI exercise1 (create a page with head, title and body); DOM-tree; AI exercise2 (add attributes to the page created in exercise1); What is CSS; Example of a CSS file; Where CSS files are stored and why (designers can update them without changing the page code); Color scheme, padding, styles, selectors; AI exercise3 (add styles to exercise2); What is JS; How to include JS (<script>); Selecting DOM & handling events; Example of a JS code; When JS is working (addEventListener, querySelector); AI exercise4 (add a button to the exercise3). 3. WebUI testing: Broken links; Cross-browser problems; Responsive design problems; How to localize these UI problems using DOM-tree; DevTools and how to use them to find WebUI bugs; JS bugs; Validation, performance, security problems; Mockups and UI components testing of a web page; Figma; How to compare Figma mockup with a real page; AI exercise5 (compare mockup with a real page, find all the differences).
Practical workshop	15 students solve real IT tasks, while a lead mentor with 10+ years of IT experience guides and assists them: - Work with dev tools to understand what components (HTML, CSS, JS) are needed for the correct display of web pages - UI testing, compare with mockups - How JS script is executed

Homework	Flight Tickets Booking application. GUI testing to cover: - A basic search for a flight as a guest - Sign-up (register a user account) - Search for a flight as a registered user

training block

WebUI Tools and Testing (W2)

Theory (textbook 18) + quiz	[All defects must be based on examples of defects from W1; no new ones should appear] 1.DevTools: Elements and What bugs can be caught with it; Exercise1 (find elements on the page); Console and What bugs can be caught with it; Exercise2 (reproduce a W1 error and copy console log); Sources and What bugs can be caught with it; Exercise3 (open Sources tab and locate a file related to an error (from W1); Network and What bugs can be caught with it; Exercise4 (find requests/responses from mentorpiece.education); Exercise5 (Network throttling / Offline mode); Application and What bugs can be caught with it; Exercise6 (clear cache and cookie). 2. Proxy-servers: What are they and why does a tester need it; The most popular proxies (Charles, Fiddler) and API-client with proxy functionality (Postman); How to configure Charles for mobile API testing (Setup device Wi-Fi proxy, Trust the Charles SSL certificate on the device). 3. Screenshot tools: Why does a tester need them; Required features of a good screenshot tool for QA; Popular screenshot tools; Embedded screenshoters in Windows, Linux, MacOS; Screen record; How to take screenshots on mobile (Android, iOS).
Practical workshop	15 students solve real IT tasks, while a lead mentor with 10+ years of IT experience guides and assists them: - Work with dev tools (elements, network, storage) - Working with postman proxy servers
Homework	Basic: Flight Tickets Booking application. GUI testing using the tools discovered in class - Book a ticket, Check booking, Cancel booking - A list of all anomalies found Advanced: - Using postman proxy (browser or system) create an API collection of the Flight Tickets Booking application.

training module

TEST DESIGN [TD]

why it's needed: to detect defects as quickly and efficiently as possible
student knows: all Test Design approaches
has the skills to: can choose the right one for a particular situation and apply it

training block

Exploratory testing (Ex1)

Theory (textbook 22) + quiz	1. What is exploratory testing. 2. Types and approaches to exploratory testing: Monkey testing, Gorilla testing, Ad hoc testing, Error guessing, Testing tours (Business District: Landmark tour, Business District: Intellectual tour, Historic District: Bad-Neighborhood Tour, Hotel District: Rained-Out Tour, Seedy district). 3. Exploratory testing sessions: Testing tours: Chrome Extension.
Practical workshop	15 students solve real IT tasks, while a lead mentor with 10+ years of IT experience guides and assists them: - Exploratory testing session with FTB and XRay app as an app for exploratory. - Explore NOT the bugs, but the main scenarios. - Script the results - Output: mind map of the FTB application
Homework	- Prepare and record a tech talk about the FTB features, investigated during the exploratory session. Each student gets a part of the application (you may use UI, DB and API for your investigation): 1) Add airport 2) Add aircraft 3) Add flight 4) Search flight 5) Verify booking 6) Login 7) All the lists

training block

Requirements Sources and Types, UML (Req)

Theory (textbook 24, 23) + quiz	1. Requirements sources: Analysts work (Customer interviews, Review of a real process, Target audience quiz, Competitor analyses, Laws and documentation analysis). 2. Types of requirements: Business requirement, User requirement, Technical requirement. 1. UML: What is it?: Requirements analysis as a QA task. 2. UML diagram types: Structure Diagrams, Behavior Diagrams. 3. UML diagrams, mostly used by QAs: Structure Diagrams, Class Diagram (Testing on the basis of it), Behavior Diagrams, Use case diagram (Testing on the basis of it), Sequence Diagram (Testing on the basis of it), State-Transition Diagram (Testing on the basis of it), UML notation cheat sheets.
Practical workshop	15 students solve real IT tasks, while a lead mentor with 10+ years of IT experience guides and assists them: - Take the req for the FTB app and API requirements. For each scenario: what services are run, what DB changes are expected. Analyze and prepare questions. Output: 1) Questions list. 2) HW (Somebody who knows the app, should answer the questions as stakeholder)
Homework	- Analyse the rest of requirements, update the mind map from the previous HW. - Analyze the requirements given in the SRS (System Requirements Specification) below. - What questions could you prepare for the authors of the requirements to clarify the business information given? How to draw a mind map?

training block

Test Documentation (TDK)

Theory (textbook 27) + quiz	1. What is Test Documentation: Advantages, Disadvantages of Test Documentation. 2. Types of Test Documentation: Checklist; Test Case (Essential info needed for a test case). 3. How to write good Test Cases: Criteria of a good test case; Example Test Cases. 4. Test Management Software (how to choose).
Practical workshop	15 students solve real IT tasks, while a lead mentor with 10+ years of IT experience guides and assists them: - Cover a functionality with some test cases - Create a test run - Execute it App: FTB Output: test case templates (2 per student)
Homework	- Google different test case templates - Ask ChatGPT about a good test case template. - Find at the internet different examples of test cases, describe them from the correctness point of view.

training block

Defects (DF)

Theory (textbook 28) + quiz	1. What is a defect and why do we need them?: Why the defect reporting is useful; Some common types of defects. 2. Defect Life Cycle. 3. How to write a good defect: More about Severity; More about Priority. 4. Defect Management System: ClickUp, Jira.
Practical workshop	15 students solve real IT tasks, while a lead mentor with 10+ years of IT experience guides and assists them: - Practice in system defects localization and reporting - Practice in defects verification App: FTB Output: defect template (2 per student).
Homework	- Write a checklist for the PUT /api/v0/flights/{id} method of http://192.168.40.100:8000/swagger-ui/index.html#/Flight - Test according to your checklist - Report defects in Jira, as it was considered in class.

training block

Static testing, Requirements Static Testing (TT_c)

Theory (textbook 29,30) + quiz	1. What is a static testing and why is it needed? 2. Static Testing Approaches: Inspection; Walkthrough; Peer review (tool-assistent) 1. Requirements Static Testing: How to assess. 2. Requirements static testing methodology on the example of business requirements: Are the requirements valuable?, Correct?, Clear?, Understandable to the target audience?, Complete?, Consistent?, Assessable?, Verifiable?, Modifiable?, Traceable?, Implementable? 3. Absence of reqs.
Practical workshop	15 students solve real IT tasks, while a lead mentor with 10+ years of IT experience guides and assists them: - Corrected test cases (pairs should be able to negotiate in DM) - Comments in TMS (with arguments!) of how to make the defects better
Homework	- Correct test cases according to review on workshop - Do the cross review of defects from HW 28 - Defects (pairs are to be prepared in advance). Don't forget to argue your comments. - Mentor review cross review

training block

Test Classification: Levels of testing (TT3)

Theory (textbook 31) + quiz	1. Levels of Testing Pyramid. 2. Unit Testing. 3. Integration Testing: Stubs and Test Drivers; Approaches to the Integration testing; Top Down approach; Bottom Up approach; Sandwich approach; What's usually happen with Integration tests. 4. System Testing: End-to-End (E2E) testing. 5. Acceptance Testing.

Practical workshop	15 students solve real IT tasks, while a lead mentor with 10+ years of IT experience guides and assists them: - Analyze the mind map, mark the branches that check the integration or add them, if do not exist - Add integration tests to the suite App: FTB Output: Integration tests included
Homework	- Cover the FTB with smoke integration tests (each level with each) - Mark the existing smoke tests with level

training block

Non-functional testing synopsys (TT4)

Theory (textbook 32) + quiz	1. The difference between functional and non-functional testing. 2. Full list of non-functional testing types. 3. Some popular non-functional testing types: UI testing; Usability testing; Security testing; Performance testing.
Practical workshop	15 students solve real IT tasks, while a lead mentor with 10+ years of IT experience guides and assists them: - Analyze the mind map, mark the branches that check the non-functional requirements or add them, if do not exist - Add NFR smoke tests to the suite - Mark "smoke", "regression", "sanity" [Removed: Test plan creation - usable for leads only) App: FTB Output: NFR smoke tests included
Homework	- Cover the FTB with smoke NFR tests (each level with each) - Mark the existing smoke tests with type (if any) - Mark "smoke", "regression", "sanity" [

training block

Test design techniques: Black, Gray, Whitebox (TT5)

Theory (textbook 33) + quiz	1. Approaches to testing: Testing pyramid; White box approach; Black box approach. 2. Boxes and testing pyramid: Unit testing vs. System/Acceptance testing; Gray box testing; Boxes combinations. 3. Approaches to Regression testing: Whitebox regression testing; Blackbox regression testing; Graybox regression testing.
Practical workshop	15 students solve real IT tasks, while a lead mentor with 10+ years of IT experience guides and assists them: - Levels and types of testing and approaches to the testing process: how to combine: Practice
Homework	You have a function that counts a square root. Mandatory: How to test this function on the unit tests/whitebox level? Create a checklist for all the checks that look useful to you. Optional (*): Write the function and unit tests for it.

training block

Test Design Techniques: Equivalence classes and Boundary Values analysis (TD1)

Theory (textbook 34) + quiz	1. Why do we need test design techniques and a bit of math: The history of test design techniques. 2. Equivalence classes; Equivalence classes partitioning use for requirements test coverage: How to partition possible airport name values into equivalence classes; How to take integration into consideration. 3. Boundary values analysis; Why is testing boundary values so important? The boundary analysis for requirements test coverage. 4. How to combine equivalence classes and boundary values for your tests. Positive and Negative tests.

Practical workshop	15 students solve real IT tasks, while a lead mentor with 10+ years of IT experience guides and assists them: - Finding where the equivalence classes and BVA could be applied in the application and why. - Train test cases writing Output: - Cases when the equivalence classes and BVA could be applied. - Test cases
Homework	- Find where the regression testing on the base of equivalence classes could be applied to your suite - Explain why here - Cover the functionality with the test cases - Prioritize the test cases and mark them with type and level

training block

Test Design Techniques: Pairwise testing (TD2)

Theory (textbook 35) + quiz	1. The idea of pairwise testing: A problem of parameters combinations; All combinations; Pairwise testing. 2. Pairwise technique use for requirements test coverage: How to use pairwise for testing flights. 3. Other Pairwise tools. 4. How to combine pairwise testing with equivalence partitioning and boundary value analysis.
Practical workshop	15 students solve real IT tasks, while a lead mentor with 10+ years of IT experience guides and assists them: - Finding where the pairwise testing could be applied in the application and why. - Train test cases writing Output: - Cases when the pairwise testing could be applied. - Test cases
Homework	- Find where the regression testing on the base of pairwise testing could be applied to your suite - Explain why here - Cover the functionality with the test cases - Prioritize the test cases and mark them with type and level

training block

Test Design Techniques: Decision tables (TD3)

Theory (textbook 36) + quiz	1. The idea of decision tables: The complex business logic problem; Steps of the table creation. 2. How to create decision tables. 3. Requirements analysis and combining Decision tables with other test design methods: A difference between a pairwise testing and decision table; A combination of decision table with equivalence classes and boundaries.
Practical workshop	15 students solve real IT tasks, while a lead mentor with 10+ years of IT experience guides and assists them: - Finding where the decision tables could be applied in the application and why. - Train test cases writing Output: - Cases when the decision table could be applied. - Test cases
Homework	- Find where the regression testing on the base of decision tables could be applied to your suite - Explain why here - Cover the functionality with the test cases - Prioritize the test cases and mark them with type and level

training block

Test Design Techniques: State-Transition diagram (TD4)

Theory (textbook 37) + quiz	1. The idea of state-transition diagrams: Objects states and transitions. 2. How to create state-transition diagram: States; Transitions. 3. How to use state-transition diagrams for testers' everyday work: How to write test scenarios based on state-transition diagrams; How to combine of state-transition diagram with equivalence classes and boundaries.

Practical workshop	15 students solve real IT tasks, while a lead mentor with 10+ years of IT experience guides and assists them: - Finding where the state-transition testing could be applied in the application and why. - Train test cases writing Output: - Cases when the state-transition testing could be applied. - Test cases
Homework	- Find where the regression testing on the base of state-transition could be applied to your suite - Explain why here - Cover the functionality with the test cases - Prioritize the test cases and mark them with type and level

training block

Test Design Realization on Real Projects (TDR)

Practical workshop	15 students solve real IT tasks, while a lead mentor with 10+ years of IT experience guides and assists them: - Interview tasks "test some new functionality" like Online cinema task. Output: understanding when and how test design methods could be applied.
Homework	Imagine that you are testing an ordinary calculator. It is a web application that can perform 4 arithmetic operations. The UI has buttons with numbers, arithmetic actions, an expression input field and a result output field. There are also ""equals"" and ""erase"" buttons. You have no requirements for the application. The task is to cover the calculator with all possible tests. The output is a checklist with checks.

training module

TEST AUTOMATION BASICS [TA]

why it's needed: to have a basic understanding of test automation (AT) and know how to start learning AT when the time is right; to work with AT tools that are often needed by manual testers as well - version control systems and virtual test environments

student knows: what TA is basically, what is Gherkin and when it applies, what is containerization and VCS

has the skills to: Postman: create easy tests using snippets; Gherkin: write tests in Gherkin format; Docker: download, start, create new container; Git: initialize a repository, make a commit, track new files, commit changes

training block

What is Test Automation and Postman as a easy entry point into TA (AU1)

Theory (textbook 38) + quiz	1. Test automation – what is that about?: Definition; Kinds of test automation 2. What to do exactly? 3. What do you need to start in Test Automation?
Practical workshop	15 students solve real IT tasks, while a lead mentor with 10+ years of IT experience guides and assists them: - A practice session with a try to understand the Postman automation code
Homework	- One method per student is covered with Postman test

training block

Test automation: Gherkin (AU2)

Theory (textbook 39) + quiz	1. Functional test automation support: Interaction between the automation team and the other team members; Behaviour Driven Development (BDD); Universal format of test cases. 2. Gherkin syntax and tools: Gherkin format; Gherkin tools. 3. Pitfalls of Gherkin.

Practical workshop	15 students solve real IT tasks, while a lead mentor with 10+ years of IT experience guides and assists them: - Try to understand the Gherkin code
Homework	- Write Gherkin code based on test cases written in previous lessons

training block

Docker (D1)

Theory (textbook 41) + quiz	1. Containerization: The difference between virtualization and containerization (in which cases a QA engineer needs the former and/or the latter; students should understand that Docker is just one example of a program that performs containerization). 2. How Docker works: A brief, high-level overview of how it works (Docker Daemon + Docker CLI/Docker UI); How to install Docker (Linux/MacOS/Windows); Basic concepts (registry, image, container); What QA engineers need in their work and why (Developers assemble images, push them to the registry, QA engineers download them from there and run containers from the images). 3. How to work with Docker, basic commands: How to build an image; How to download/push an image; How to run a container from an image, how to kill, stop, restart; How to interact with an application inside a container (port mapping); How to execute commands inside a container (in interactive mode and without); How to copy files to and from a container; Why something can be done in one container but not in another (how images are assembled, what they consist of (layers, what they inherit from the images on which the image is based)); Basic Dockerfile instructions.
Practical workshop	15 students solve real IT tasks, while a lead mentor with 10+ years of IT experience guides and assists them: - Demonstration and practice with docker containers
Homework	Basic: - Pull image; Start the container from - Check the metadata of the “nginx” image and get ... - Create <your_username>_index.html with random content and mount it to ... Advanced tasks: - Create your own image

training block

Git Elementary for QA Engineers (G1)

Theory (textbook 42.1) + quiz	1. What is Git and why QA engineers may need to know it: Overview of Git; Benefits of Git for QA processes; Why understanding Git is essential for QA engineers. 2. Key Git concepts: What is a repository; File statuses in Git (Tracking, Staging, Committing). 3. Getting started with Git: Installing; First-Time Git Setup, UI for Git 4. Basic Git commands: Initializing a repository (git init); Checking and managing file statuses (git status, git add, git reset); Committing changes (git commit). 5. Working with branches (basic operations): What is a branch; Branch structure inside a repository; Creating and switching branches using git branch, git switch, and (optionally) git checkout (legacy command still used in many repositories); Local branch workflow overview (optional brief mention that branches can be merged - details covered in G2).
Practical workshop	15 students solve real IT tasks, while a lead mentor with 10+ years of IT experience guides and assists them: - Creating a local repository and basic commands (git init, git status, git add, git commit, git reset). - Working with files in the repository (touch, git add, git commit -m, git rm, git reset). - Viewing the history of changes (git log, git show).
Homework	- Creating files - Commits - Creating, deleting and restoring a directory - Displaying the commit log

training block

Git: Working with Branches and Remote Repositories (G2)

Theory (textbook 42.2) + quiz	1. Setting up SSH and working with remote repositories: Configuring SSH for GitHub (Windows, Linux, macOS); Adding a remote repository; Cloning repositories; Difference between HTTPS and SSH connections. 2. Branches and merging in Git: Recap of basic branch operations (git branch, git switch, git checkout); Merging commits (merge vs fast-forward merge vs rebase); Resolving merge conflicts; Working with remote branches. 3. Syncing data between local and remote repositories: Commands for syncing (fetch, pull, push); Understanding how fetch and pull differ. 4. How Git supports QA processes: Brief about code review; Continuous Integration (CI); Continuous Deployment (CD).
Practical workshop	15 students solve real IT tasks, while a lead mentor with 10+ years of IT experience guides and assists them: - Working with branches (git checkout -b, git checkout, git merge, git branch -D, git branch -m). - Working with remote repositories (git remote add, git push, git clone). - Synchronization and resolving conflicts (git pull, git push, resolving conflicts manually, git merge)
Homework	- Creating a public (open) repository on github.com - Cloning a repository - Working with branches - Commit, merge, push

training module

SOFTWARE LIFECYCLE AND DEVELOPMENT PROCESS [Ag]

why it's needed: to work productively in a team with other QAs and communicate effectively with developers and customer stakeholders
student knows: what is Agile and what are its benefits, scram activities, SRLC
has the skills to: participate in planning, standups and retrospectives, not to get into deadlock situations on the project (the rule "if you don't know how to solve a problem for 2 hours, ask", etc.)

training block

Agile vs. Waterfall (A_c)

Theory (textbook 25) + quiz	1. Agile versus Waterfall method of software development: Advantages, Disadvantages of Waterfall; Advantages, Disadvantages of Agile. 2. Key principles and concepts of Agile: The Agile Manifesto. 3. Type of internships (startups vs big companies, QC expierence vs QA expirience, why startups are better)
Practical workshop	15 students solve real IT tasks, while a lead mentor with 10+ years of IT experience guides and assists them: - Business game: One sprint in 2 hours: practice with backlog grooming and sprint planning OR - Analyze the board of an internship customer (if all the accesses are granted) Tasks decomposition and statuses
Homework	- Analyse the user story below and 3 attachments to it. - Create a set of questions for the hypothetical grooming, where you could ask questions to analyst team.

training block

SCRUM (A2)

Theory (textbook 26) + quiz	1. Agile Implementation and Methodologies: Major Forms of Agile; Kanban, Lean; SCRUM. 2. SCRUM: Roles used in a SCRUM implementation of Agile; SCRUM Terminology; SCRUM meetings. 3. Project Management Software: User Story.

Practical workshop	15 students solve real IT tasks, while a lead mentor with 10+ years of IT experience guides and assists them: - Work with user stories in Jira - Practice in standups - Practice in retrospective - Practice in sprint demo OR - Configure the process of meetings for the internship
Homework	- Estimate testing of the story in story points. - How does your estimation affect the team velocity? - Think about potential problems and prepare the description of these problems - How the story could be demonstrated to stakeholders? - RECORD a speech about the work done and your plans for a daily meeting

training block

Releases (Rel1)

Theory (textbook 40) + quiz	1. Software release life cycle (SRLC) with the episodes from "Silicon Valley": What is the release life cycle; Before the main developement starts: POC; Before the main developement starts: MVP; Pre-Alpha version; Alpha version; Beta version; Release Candidate; Release (Production); Maintenance stage. 2. Feature and code freeze for a release: Feature freeze; Code freeze; Release branch cut. 3. Environments for release implementation and testing: Development and QA environments; Staging and Live.
Practical workshop	15 students solve real IT tasks, while a lead mentor with 10+ years of IT experience guides and assists them: - Analyze the test environments and releases of a customer
Homework	Answer the questions: - Today is a feature freeze, but on a daily standup, developers ask you to verify a defect because they should fix it before the release. Will you verify or wait for the next release? - Today is a code freeze, and one of the developers promises to fix a defect by the end of the day. Should you verify this defect right after he fixes it or this task is not that critical?

training module

AI-QA-Engineer [AI]

why it's needed: testing AI applications is the immediate future of the QA engineer profession.
student knows: fundamental differences between AI applications from a testing perspective, their development and testing lifecycle, types of AI applications, and their functional/non-functional testing
has the skills to: black-box testing of AI applications—both manually and using vibe coding

training block

AI applications testing: Non-LLM metrics (AI.J.1)

Theory (textbook) + quiz	1. What is AI: History of AI development; Types of AI: Strong vs Weak AI, ML, DL; How AI works; Fundamental differences between AI systems and classic software (Non-determinism, Absence of source code for logic, Performance degradation); Leading AI providers at present (Anthropic, Google, OpenAI, Microsoft). 2. AI testing from a QA perspective: Using AI tools for testing classic apps vs testing AI-apps; AI Development & Testing Lifecycle (Data collection, Training, Inference, Monitoring in production); The role of QA at different stages; 5 types of AI-applications (Semantic search, Classification, Clustering, Regression, Generation (Text, Code, Image)); Black/Gray Box testing. 3. LLM vs Non-LLM metrics; Non-LLM metrics for Semantic Search metrics (Precision, Recall, F1-Score, RR, MRR) [Sim AI1.30,31,33,34].
Practical workshop	15 students solve real IT tasks, while a lead mentor with 10+ years of IT experience guides and assists them: - Calculating the Non-LLM metric using the output of a real AI application
Homework	Calculation of Non-LLM Metrics

training block

AI applications testing: LLM metrics (AI.J.2)

Theory (textbook) + quiz	1. Prompt engineering: How to create prompts (Elements of a prompt: Persona, Task, Format, Context) [Sim AI2.10,11]; Prompts Repeatability, Temperature [Sim AI2.25,26]. 2. LLM Metrics; LLM Metrics for Generation (Relevancy, Accuracy, Format Accuracy, Faithfulness, Clarity, Hallucinations [Sim AI2.30-35]); Other LLM metrics: Classification metrics (Recall, F1-Score); How QA utilizes them to assess model quality. 3. Adversarial Testing: System prompt vs User Prompt [Sim AI2.40]; Methods (Direct/Indirect Prompt Injections [Sim AI2.41-42], Jailbreaking (DAN) [AI2.43]; Prompt injections preventing). 4. Vibe Coding: What is it; Why it is need for QA (code templates generation for automation); Task decomposition, generation, debugging; Pros and cons (Limitations; Need for Verification); Why coding is not programming (Programming is design, architecture, planning, coding, testing - Vibe-coding focuses only on the coding phase); Claude Code Setup & Usage (Practical execution in VS Code); GitHub Codespaces + GitHub Copilot; Standard code mode vs Agent mode. Your first vibe-created app.
Practical workshop	15 students solve real IT tasks, while a lead mentor with 10+ years of IT experience guides and assists them: - Analyzing complex cases of calculating Relevancy, Accuracy, Faithfulness, Clarity, and Hallucinations. - Performing successful jailbreaking using the DAN method. - Performing vibe coding and analyzing its results.
Homework	- Completing exercises in Mentorpiece Sim Advanced: - Refining the app using Vibe coding

training block

AI applications testing: Non-functional AI testing (AI.J.3)

Theory (textbook) + quiz	1. AI Non-functional testing: Costs testing; Traceability testing; Reliability testing (Latency [Sim AI3.10], Throughput, Robustness (resilience to "noise" in data)); Privacy and data leakage testing). 2. For further study: LLM-as-a-Judge testing; How to choose the right AI-model (Benchmarks, Their pros and cons; DeepEval); AI-Agents; AI-Agents testing; RAG; RAG testing [Sim AI3.30]; Fine-tuning; Data testing; Gray box testing. 3. Costs testing: Your first AI-autotest. 4. Useful resources for further study (free: Learnpython.org, HuggingFace; AI-QA-Engineer White Book).
Practical workshop	15 students solve real IT tasks, while a lead mentor with 10+ years of IT experience guides and assists them: - Analyzing complex cases of non-functional testing calculations - Writing automated tests using vibe coding
Homework	Designing and refining automated tests using Vibe coding

why it's needed: to summarize the results of the in-house education and get the internship goals

Graduation (GR)

Practical workshop	1. What the every student has learned [review 60.1c for every student] 2. Feedback from every student about education: not good/good/stop/evolve 3. Feedback from every mentor to the whole group: quiz fulfillment/WS activity/HW wit by every module (NEED MENTORS PREP!) 4. INT + Exam = 40% of Attestation grade! 5. Goals for the internship (1st month 2 grades, 2d month 2 grades) 6. Reminder of the final exam-interview, the need to repeat theory even during the internship 7. "Advanced troubleshooting for QA" schedule 8. Alumni: tips on finding a job 9. Guest internship customer

INTERNSHIP [INT]

why it's needed: a) to turn knowledge into skills b) get an idea of how QA works on a real business project c) get work experience for resume
student knows: what knowledge to use and when to use it
has the skills to: perform QA work on a variety of tasks and bring value to both the team and the business

8 sprints on a business project (4 months)	Every sprint: 1. Backlog preparation 2. Sprint planning 3. Daily meetings 4. Studying customer requirements and application architecture 5. Verification and validation of customer applications 6. Defect verification 7. Regression testing 8. Testing new functionality 9. Communication with the customer's team 10. Sprint retrospective

INTERSTATE 60 [I60]

FINAL INDIVIDUAL EXAM-INTERVIEW [INT]

II. TERMS OF STUDY

module	code	module weight *	min. number of points**	max. number of points
ENTRANCE (entrance testing, vocational guidance, entrance exam)	ENT	2.94%	n/a	n/a
Relational databases for QA [DB]	DB	5.88%	5	9
LINUX FOR QA [L]	L	5.88%	9	15
NETWORKS FOR QA [N]	N	5.88%	12	21
SOFTWARE ARCHITECTURE FOR QA [Tr]	Tr	5.88%	16	26
REST & API FOR QA [R]	R	5.88%	19	32
WEB UI FOR QA [W]	W	5.88%	23	38
TEST DESIGN [TD]	TD	5.88%	26	44
TEST AUTOMATION BASICS [TA]	TA	5.88%	30	50
SOFTWARE LIFECYCLE AND DEVELOPMENT PROCESS [Ag]	Ag	5.88%	34	56
AI-QA-ENGINEER [AI]	AI	5.88%	37	62
INTERNSHIP [INT]	INT	17.65%	48	79
INTERSTATE 60 [I60]	I60	2.94%	49	82
FINAL INDIVIDUAL EXAM-INTREVIEW [INT]	Fin	17.65%	60	100

* The weight of the assessment for the training module in terms of the certification assessment.
** The minimum current value of the Assessment Score after the Hard Deadline for the last assignment in a specific module, at which the student can continue their studies.

CONDITIONS FOR EXPULSION

- **Either the current value of the Assessment Score after the Hard Deadline for the last assignment in the current module is below the minimum value (see table above)**
- **Or two consecutive learning blocks (quiz + workshop + homework) with strict deadlines have a score below 30.**

GRADUATION AND EMPLOYMENT

1. **Successful completion of the course is considered to be a certification score of 60 points or higher.**
2. **To participate in individual employment mentorship, you must have a current certification score of 60 points or higher.**
3. **Students with a certification score of 90 or higher receive:**

- **Confirmation of knowledge on LinkedIn from three mentors**
- **A letter of recommendation from the lead mentor about the results of work on an IT project (internship). It is recommended to include a link to this in your resume.**