Національний технічний університет України
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Department of Computer Systems Software

Web Application Development

Work program of the discipline (Syllabus)

Details of the discipline

Level of higher education	*First (bachelor's)*
Field of knowledge	*12 Information Technology*
Speciality	*121 Software engineering*
Educational program	*Software Engineering of Multimedia and Information Retrieval Systems*
Discipline status	*Elective*
Form of study	*Full-time*
Year of preparation, semester	*3rd year of preparation, 5th semester*
Scope of discipline	*4 ECTS credits / 120 hours. Lectures: 36 hours, practical work: 18 hours, individual work: 66 hours.*
Semester control / control measures	*Pass/fail credit, module tests, calendar control*
Timetable	*According to the schedule for the autumn semester of the current academic year (rozklad.kpi.ua)*
Language of instruction	*English*
Information about Course Leader / Instructors	*Lectures and practical classes are conducted by:* *Associate Professor Andrii V. Petrashenko, PhD, petrashenko@gmail.com*
Course Placement	*Google Classroom. Materials on the discipline "Web Application Development"*

The program of the discipline

1. Description of the discipline, its purpose, subject of study and learning outcomes

The discipline "Web Application Development" introduces students to the fundamental principles and technologies used to create interactive web applications. The course follows the complete path from a browser interface to a small application that communicates with a server and stores persistent data.

***The purpose** of studying the discipline is to develop students' ability to understand how web applications work and to create small applications using HTML, CSS and JavaScript in the browser together with a simple Node.js / Express backend and SQLite data storage.*

***The subject** of the discipline is the design and development of interactive web applications, including browser-side interfaces, HTTP communication, basic server-side processing, web APIs and persistent application data.*

The study of the discipline forms students' general competencies (GC) and professional competencies (PC) necessary to solve practical problems of professional activity:

***GC02** Ability to apply knowledge in practical situations.*

***PC03** Ability to develop architectures, modules and components of software systems.*

***PC07** Knowledge of data information models, ability to create software for data storage, extraction and processing.*

***PC10** Ability to accumulate, process and systematize professional knowledge and recognize the importance of lifelong learning.*

PC12 *Ability to carry out system integration and apply procedures that maintain the integrity, functionality and reliability of software.*

PC13 *Ability to reasonably choose and master software development and maintenance tools.*

Program learning outcomes (PLO) supported by the discipline according to the educational program:

PLO01 *To analyze, purposefully search for and select information and reference resources necessary for solving professional tasks, taking into account modern achievements of science and technology.*

PLO07 *To know and apply in practice the fundamental concepts, paradigms and basic principles of the functioning of computing and software engineering tools.*

PLO12 *To apply efficient approaches to software design in practice.*

PLO14 *To apply in practice tools for software design, testing, visualization, measurement and documentation.*

PLO15 *To reasonably choose programming languages and development technologies for creating and maintaining software.*

PLO18 *To know and apply information technologies for data processing, storage and transmission.*

PLO23 *To be able to document and present the results of software development.*

As a result of studying the discipline, students are expected to be able to explain the basic architecture of a web application; create structured and responsive interfaces with HTML and CSS; implement browser-side behavior using JavaScript, DOM events and forms; manage simple application state; use localStorage; exchange JSON data with web APIs using fetch(); create a simple Node.js / Express server and API; store application data in SQLite; connect frontend, backend and persistent storage; perform basic validation, debugging and error handling; and recognize common web security risks.

2. Prerequisites and post-requisites of the discipline (place in the structural and logical scheme of training in the relevant educational program)

Successful study of the discipline requires basic programming skills and familiarity with fundamental HTML and CSS concepts. Students are expected to understand variables, conditions, loops, functions and basic data structures. Previous JavaScript experience is useful but the necessary concepts are reviewed and applied during the course.

The knowledge and skills acquired in the course can be used in further study and professional activities related to frontend development, backend development, software architecture, cloud technologies and full-stack web application development.

3. The content of the discipline

The discipline "Web Application Development" involves the study of the following topics:

Topic 1. Web application foundations: web pages and web applications, browser and server roles, URLs, client-server architecture and the HTTP request-response model.

Topic 2. HTML and CSS for web applications: semantic HTML, forms, CSS layout, Flexbox, Grid and responsive design basics.

Topic 3. JavaScript in the browser: JavaScript fundamentals for web development, DOM, events and dynamic user interfaces.

Topic 4. Forms and application state: user input, client-side validation, objects and arrays, rendering state, CRUD operations in the browser and localStorage.

Topic 5. HTTP, JSON and external web APIs: HTTP methods and status codes, JSON, asynchronous JavaScript, promises, async/await and fetch().

Topic 6. Server-side web development: Node.js, npm, Express, routes, request/response processing, static files and simple web APIs.

Topic 7. Persistent data: SQLite, simple tables, basic data operations and connecting an Express application to persistent storage.

Topic 8. Full web application integration: browser-to-server communication, frontend-backend data flow, validation and error handling.

Topic 9. Web application quality and security: browser developer tools, server logs, basic testing concepts, untrusted input, XSS and injection concepts, authentication and authorization overview, application structure and basic deployment concepts.

Module tests

Semester control: pass/fail credit

4. Training Materials & Resources

Basic literature

1. *Springer S. Node.js: The Comprehensive Guide / Sebastian Springer. - Rheinwerk Computing, 2022. - 834 p. - ISBN 978-1-4932-2292-6.*
2. *MDN Web Docs. HTML, CSS, JavaScript, HTTP and Web APIs [Electronic resource] / Mozilla. - Available at: https://developer.mozilla.org/. - Accessed: 30.12.2024.*

Further reading

1. *Node.js Documentation [Electronic resource]. - Available at: https://nodejs.org/docs/. - Accessed: 30.12.2024.*
2. *Express Documentation [Electronic resource]. - Available at: https://expressjs.com/. - Accessed: 30.12.2024.*
3. *SQLite Documentation [Electronic resource]. - Available at: https://www.sqlite.org/docs.html. - Accessed: 30.12.2024.*

Educational content

5. Methods of mastering the discipline (educational component)

No.	Type of training session	Description of the training session
1	*Lecture 1.*	*Introduction to Web Applications. Web page vs web application; browser, server, URL, HTTP request-response model; roles of HTML, CSS and JavaScript; TaskBoard as the recurring example.* *Self-study: preparation for Lecture 1.*
2	*Lecture 2.*	*HTML and CSS for Web Applications. Semantic HTML, forms, CSS selectors, the box model, Flexbox, Grid and responsive layout basics.* *Self-study: preparation for Lecture 2.*
3	*Practical work 1.*	*Web Application Interface. Create the basic HTML/CSS interface of the individual application using semantic structure, forms and a simple responsive layout.* *Self-study: preparation and completion of Practical work 1.*
4	*Lecture 3.*	*JavaScript in the Browser. JavaScript values, variables, functions, arrays and objects in the context of browser applications.* *Self-study: preparation for Lecture 3.*
5	*Lecture 4.*	*DOM and Events. Selecting and modifying elements, event listeners, dynamic content and simple UI behavior using modern browser APIs.* *Self-study: preparation for Lecture 4.*

No.	Type of training session	Description of the training session
6	Practical work 2.	Interactive Web Page. Add dynamic behavior to the individual application using JavaScript, DOM manipulation and browser events. Self-study: preparation and completion of Practical work 2.
7	Lecture 5.	Forms and Validation. Reading user input, form submission, preventing default behavior, validation rules, error messages and user feedback. Self-study: preparation for Lecture 5.
8	Lecture 6.	Application State and Browser Storage. Rendering data from arrays/objects, add/edit/delete operations, application state and localStorage. Self-study: preparation for Lecture 6. Module test 1.
9	Practical work 3.	Forms, Data and Browser Storage. Implement create/edit/delete operations, validate input and preserve application data using localStorage. Self-study: preparation and completion of Practical work 3.
10	Lecture 7.	HTTP, JSON and Web APIs. HTTP methods and status codes, request/response data, JSON and the role of APIs in web applications. Self-study: preparation for Lecture 7.
11	Lecture 8.	Asynchronous JavaScript and Fetch. Promises, async/await, fetch(), loading states, error handling and working with external APIs. Self-study: preparation for Lecture 8.
12	Practical work 4.	Working with a Public API. Retrieve data from an instructor-approved public API, display it in the application and handle loading/error states. Self-study: preparation and completion of Practical work 4.
13	Lecture 9.	Introduction to Node.js. Browser JavaScript vs server-side JavaScript, Node.js runtime, modules, packages and npm. Self-study: preparation for Lecture 9.
14	Lecture 10.	Express and the First Web Server. Express application, routes, request and response objects, static files and JSON responses. Self-study: preparation for Lecture 10.
15	Practical work 5.	First Node.js and Express Application. Create and run a simple Express server, serve the application and implement basic routes. Self-study: preparation and completion of Practical work 5.
16	Lecture 11.	Designing a Simple Web API. Resources and endpoints; GET, POST, PUT/PATCH and DELETE; route parameters, query parameters and request bodies. Self-study: preparation for Lecture 11.
17	Lecture 12.	Frontend-Backend Communication. Calling a custom API from the browser with fetch(), sending and receiving JSON, status codes and API errors. Self-study: preparation for Lecture 12.
18	Practical work 6.	Building a Simple Web API. Implement basic API operations for the individual application using Node.js and Express. Self-study: preparation and completion of Practical work 6.
19	Lecture 13.	Persistent Data and SQLite. Why server memory is not enough, persistent storage, SQLite, simple tables and basic SQL operations in a web application. Self-study: preparation for Lecture 13.
20	Lecture 14.	Using SQLite from a Web Application. Connecting Express to SQLite; simple SELECT, INSERT, UPDATE and DELETE operations; handling database results and errors. Self-study: preparation for Lecture 14.
21	Practical work 7.	Persistent Web Application. Store the individual application data in SQLite and access it through the Express backend. Self-study: preparation and completion of Practical work 7.
22	Lecture 15.	Connecting the Complete Application. Browser -> fetch() -> Express -> SQLite -> JSON -> Browser. Data flow through the complete application. Self-study: preparation for Lecture 15. Module test 2.

No.	Type of training session	Description of the training session
23	*Lecture 16.*	*Errors, Validation and Basic Web Security. Server-side validation, HTTP errors, untrusted input, XSS and injection concepts; authentication and authorization overview.* *Self-study: preparation for Lecture 16.*
24	*Practical work 8.*	*Frontend + Backend Integration. Connect the browser interface to the Express/SQLite backend and add server-side validation and basic error handling.* *Self-study: preparation and completion of Practical work 8.*
25	*Lecture 17.*	*Testing and Debugging Web Applications. Browser DevTools, Network panel, console, server logs, manual testing of application flows and introductory automated testing concepts.* *Self-study: preparation for Lecture 17.*
26	*Lecture 18.*	*Web Application Architecture and Review. Client/server responsibilities, separation of concerns, application structure, maintainability, configuration and basic deployment concepts.* *Self-study: preparation for Lecture 18.*
27	*Practical work 9.*	*Final Web Application. Complete, demonstrate and explain the individual web application that integrates HTML/CSS/JavaScript, fetch(), Node.js/Express and SQLite.* *Self-study: preparation and completion of Practical work 9.*

Module tests

Total classroom workload: 54 hours (18 lectures x 2 academic hours = 36 hours; 9 practical works x 2 academic hours = 18 hours).

6. Independent work of a student/graduate student

The discipline is based on independent preparation for classroom classes, reading of current web development documentation, completion and refinement of practical tasks, debugging of the individual application, defence of practical works and preparation for module tests and semester control. During lectures, the instructor uses a small TaskBoard application as a recurring example. Each student develops an individual application domain of comparable complexity using the same technical progression.

No.	Name of the topic to be submitted for self-study	Number of hours	Literature
1	*Preparation for Lecture 1*	*1*	*1, 3*
2	*Preparation for Lecture 2*	*1*	*1, 3*
3	*Preparation for Lecture 3*	*1*	*1, 3*
4	*Preparation for Lecture 4*	*1*	*1, 3*
5	*Preparation for Lecture 5*	*1*	*1, 3*
6	*Preparation for Lecture 6*	*1*	*1, 3*
7	*Preparation for Lecture 7*	*1*	*1, 3*
8	*Preparation for Lecture 8*	*1*	*1, 3*
9	*Preparation for Lecture 9*	*1*	*2, 4-5*
10	*Preparation for Lecture 10*	*1*	*2, 4-5*
11	*Preparation for Lecture 11*	*1*	*2, 4-5*
12	*Preparation for Lecture 12*	*1*	*2, 4-5*
13	*Preparation for Lecture 13*	*1*	*2, 4-6*
14	*Preparation for Lecture 14*	*1*	*2, 4-6*
15	*Preparation for Lecture 15*	*1*	*2, 4-6*
16	*Preparation for Lecture 16*	*1*	*2, 4-6*
17	*Preparation for Lecture 17*	*1*	*1-6*
18	*Preparation for Lecture 18*	*1*	*1-6*
19	*Preparation, completion and defence of Practical work 1*	*4*	*1, 3*
20	*Preparation, completion and defence of Practical work 2*	*4*	*1, 3*
21	*Preparation, completion and defence of Practical work 3*	*4*	*1, 3*
22	*Preparation, completion and defence of Practical work 4*	*4*	*1, 3*

No.	Name of the topic to be submitted for self-study	Number of hours	Literature
23	Preparation, completion and defence of Practical work 5	4	2, 4-5
24	Preparation, completion and defence of Practical work 6	4	2, 4-5
25	Preparation, completion and defence of Practical work 7	4	2, 4-6
26	Preparation, completion and defence of Practical work 8	4	2, 4-6
27	Preparation, completion and defence of Practical work 9	4	1-6
28	Preparation for module tests	6	1-6
29	Preparation for semester control / pass/fail credit	6	1-6

Total individual work: 66 hours.

Policy & Control

7. Academic discipline policy (educational component)

Attending classes. *Absence from the classroom does not automatically result in penalty points, since the final rating is formed from assessed learning outcomes. Active participation, discussion of technical decisions and defence of practical work are evaluated when they are part of the corresponding control measure.*

Practical work and individual variants. *Practical works form a cumulative individual web application project. The TaskBoard application demonstrated during lectures is a teaching example. Students develop an instructor-assigned or approved subject domain of comparable complexity. Direct reproduction of the complete lecture application without adaptation to the individual variant does not satisfy the requirements of the practical work.*

Missed assessment controls and appeals. *A student has the right to make up missed assessment activities for a valid reason through an agreed individual procedure. Students may reasonably challenge assessment results and receive an explanation of the applied criteria in accordance with university procedures.*

Academic integrity. *The principles of academic integrity are defined by the Code of Honor of the National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute" (https://kpi.ua/code). Submitted work must represent the student's own understanding and results. Sources and borrowed code fragments must be acknowledged where appropriate.*

Use of generative AI and external tools. *Generative AI tools, documentation, forums and code assistants may be used for learning and troubleshooting when permitted by the instructor. The student remains responsible for understanding, verifying and explaining every submitted technical decision and code fragment.*

Security and external services. *Students should use only instructor-approved public APIs and services. Sensitive personal information, private credentials and payment information must not be included in practical work. Server-side input must be treated as untrusted data.*

Norms of ethical behavior. *The norms of ethical behavior of students and employees are defined by the Code of Honor of Igor Sikorsky Kyiv Polytechnic Institute.*

Inclusive education. *Course materials and assessment procedures may be adapted for students with special educational needs in accordance with university procedures and the technical requirements of the discipline.*

Teaching in a foreign language. *The language of instruction is English; students are expected to work with English-language web documentation and development tools.*

8. Types of control and rating system for assessing learning outcomes (CRO)

During the semester, students complete 9 practical works and 2 module tests. The maximum semester rating is 100 points: practical works - 60 points; module tests - 40 points.
Practical works No. 1-8 are worth up to 6 points each. They progressively develop the student's individual web application from a browser interface to an application that communicates with a Node.js / Express backend and stores data in SQLite.
For Practical works No. 1-8, points are awarded for:
- completion of the required functionality and evidence of the obtained result: 0-3 points;

- correct adaptation to the individual application and completion of the instructor's additional task or verification activity: 0-2 points;
- ability to explain the implemented solution and the role of the used web technologies: 0-1 point.
Assessment criteria for Practical works No. 1-8: 6 points - the required functionality works correctly, the individual task is completed, appropriate evidence is provided, and the student clearly explains the result; 5 points - the work is generally correct with minor shortcomings; 3-4 points - the main functionality is implemented, but part of the task or explanation is incomplete; 1-2 points - only part of the required work is implemented and/or the student cannot sufficiently explain the result; 0 points - the work is not completed or no verifiable result is provided.
Starter files, code fragments or partial examples provided by the instructor may be used as a basis for practical work, but the student must complete the required changes for the individual variant and be able to explain the submitted result.
Practical work No. 9 - Final Web Application - is worth up to 12 points. The student demonstrates a completed individual application with an HTML/CSS/JavaScript frontend, communication with a Node.js / Express backend using HTTP and JSON, SQLite persistent storage, and basic validation and error handling.
Practical work No. 9 is evaluated as follows: functional completeness - 0-4 points; correct integration of frontend, backend and persistent storage - 0-3 points; code structure, validation and handling of basic error situations - 0-2 points; demonstration and explanation of the application - 0-3 points.
Maximum number of points for practical works: 6 points x 8 practical works + 12 points x 1 final practical work = 60 points.
During the semester, students complete 2 module tests using Google Forms. Each module test is worth up to 20 points and covers the concepts and practical principles studied in the corresponding part of the course.
Module tests consist of objective questions and may include browser and server responsibilities, HTML/CSS/JavaScript behavior, DOM and events, forms and validation, HTTP methods and status codes, JSON and fetch(), Node.js and Express, API endpoints, SQLite persistence, frontend-backend interaction, basic security and debugging.
Each module test is scored in Google Forms according to the number and weighting of correct answers. If the internal Google Forms score uses a different scale, the result is converted proportionally to a maximum of 20 points.
Maximum number of points for module tests: 20 points x 2 module tests = 40 points.
The rating scale for the discipline is: R = Rpractical + Rmodule = 60 points + 40 points = 100 points.
Calendar control is carried out twice during the semester. At the first calendar control (8th week), a student receives "passed" if the current rating is at least 25 points. At the second calendar control (14th week), a student receives "passed" if the current rating is at least 45 points.
Semester control: pass/fail credit.
Conditions for admission to semester control: completion and defence of all mandatory practical works. A student with a semester rating of at least 60 points receives a passing grade automatically according to the university rating scale. A student who does not receive an automatic passing grade or wishes to improve the result may complete an additional semester assessment according to the procedure announced by the instructor; the better valid result is used.

Table of correspondence of rating points to grades on the university scale:

Score	Grade
100-95	*Excellent*
94-85	*Very good*
84-75	*Good*
74-65	*Satisfactory*
64-60	*Sufficient*
Less than 60	*Unsatisfactory*
Admission conditions are not met	*Not allowed*

9. Additional information on the discipline (educational component)

Work program of the discipline (syllabus):

Prepared by A. V. Petrashenko, PhD, Associate Professor, Department of System Programming and Specialized Computer Systems.

Approved by the Department of Computer Systems Software (Minutes No.12 of 11.02.2026)

Approved by the Methodological Commission of the Faculty of Applied Mathematics (Minutes No.7 of 03.02.2026)