Last Updated on - September 23, 2021

Course Outline - Minor Degree in Advanced Web Development

The Minor degree programme is divided into three courses:

1. Web Development 101 - Getting Started with Javascript
2. Web Development 201 - Server-side programming with Ruby on Rails
3. Web Development 301 - Typed Functional programming

These three courses along with the Minor Project or Industry Internship that together adds to 20 Credits.

Minor Degree in Advanced Web Development

Course Structure						
			Credits			
Sl	**Course Code**	**Title**	**L**	**T**	**P**	**Total**
1	WD101	**Web Development 101** Getting Started with JavaScript	0	1	0	1
2	WD201	**Web Development 201** Server Side Programming with Ruby on Rails	0	6	0	6
3	WD301*	**Web Development 301** Typed Functional Programming	0	6	0	6
4	WD401**	**Minor Project/Industry Internship** Project based experience	0	0	7	7
TOTAL			**0**	**13**	**7**	**20**

The course outlines for WD101 and WD201 courses as approved by AICTE under NEAT programme are described below. WD301* is provisionally approved by AICTE under NEAT programme. An updated version of the WD301 is being prepared and details will be shared with the Institutes later. Details on WD401** will be shared once the approach is approved by AICTE under NEAT programme.

The evaluation scheme for WD101 and WD201 courses are described in the latter part of this document.

Web Development 101 - Getting Started with Javascript

Course Objectives

This course is meant for students who do not have prior programming experience, or have a light background, and are looking to build a robust foundation for computational thinking.

They'll learn to deconstruct what software applications do, and reason about the essence of computation as transformation of data from one shape to another.

Practically, they will be able to set up a development environment, be introduced to HTML & CSS, and learn to program in a functional subset of JavaScript.

By the end of the course, they will build a 2D game that runs on the browser and be able to manipulate the DOM as well as the canvas element, and have the ability to create and deploy a basic website to the internet.

Prerequisites

- Students should have access to a computer with a modern OS (Windows 10 or above, Ubuntu 20.04 and above, macOS 10.15 and above).
- Students should have computer literacy. It includes skills like the ability to browse the internet and find information, ability to use software applications like word processors and spreadsheets, and be comfortable with user-level operating system concepts like CPU, memory, disks, and files and folders.

Course Outcomes

By completing the WD 101 course, students will gain a foundation in programming and computational thinking, and be introduced to the field of web development.

Specifically, they will learn how to:

- Set up a development environment.
- Create and style basic web pages.
- Transform data with JavaScript.
- Use computational abstractions.

- Work with the HTML Canvas.
- Create a computational model for a game.
- Use side-effects to render game state as graphical output in Canvas.

Course Content

Level 1 - Welcome to the course

Introduction to Web

- The Internet
- Client & Server
- IP address and URL
- The World Wide Web (WWW)

Set up your developer environment

- Installing Visual Studio Code
- Installing the Prettier VSCode extension

Running Linux inside Windows

- Install Ubuntu in Windows, using WSL
- Install Ubuntu using virtual machine software

Level 2 - Let's create our own websites!

Let's create our own websites

- You can make and host your very own website today!
- Our first HTML page - in about 10 minutes
- The <img> tag - let's show an image in our page
- Looking inside websites using "Inspect Element"

Deploy

- Deploy! Put your website on the internet, and get a URL you can share with the world

Level 3 - Style Matters

A quick walkthrough!

- Styling HTML using CSS files

- Learn more about CSS

Tailwind CSS

- Introduction to Tailwind CSS
- Working with Tailwind
- Tailwind's relationship with JavaScript

Level 4 - Working with Strings

Working with Strings

- Introduction to strings
- Joining strings together

Switching to the VSCode editor

- Putting HTML and JS together
- Adding comments to HTML and JS

Continue playing with strings

- Find the length of a string
- Search for a string inside another string
- String equality comparison
- Sort a collection of strings
- Split strings by a pattern

Level 5 - Numbers, Booleans, Objects and Arrays

Number Data Type

- Numbers

Boolean Data Type

- Boolean - comparisons and logical operations

Object Literals

- Object Literals - create, read & update + nesting objects

Arrays

- Arrays - handling ordered values

Level 6 - Functions - code we can call multiple times

Function

- Functions - Part 1
- Functions - Part 2
- Explicitly return a value from a function
- Passing a function as an argument

Milestone Task: Create a name to initials converter function

Level 7 - Collections

Iterating over Arrays

- Iterating over an array using the forEach method
- Generate an HTML list from an array
- Using the index of the array value during iteration
- Nested Array iteration

Transforming Arrays

- Transforming from one array to another using the map method
- Generate an HTML list from an array using the map function
- Using index of array value with map
- Transforming Nested Arrays

Filtering Arrays

- Filter an array based on some criteria
- A minimal UI for filtering flight search results
- Use the index of the array value with filter

Milestone Task: Create a function to filter data

Level 8 - Building a game with Canvas

Intro & Setup

- Why should you learn about the HTML canvas element?
- Install Node.Js on your PC and serve up some HTML & JS

Getting Started

- First Steps with Canvas
- Dynamic Drawings

The Beginnings of a Game

- Martial Arts Sim - Intro
- Loading External Images
- Using Callbacks
- Animating with Images
- Loading Multiple Animations
- Taking Control

Capstone Project - Build a personal website

Web Development 201 - Server-side programming with Ruby on Rails

Objective

The objective of this course is to teach students how to build web applications using the Ruby on Rails framework, with focus on industry-practices like object-oriented design, programming style guides, security, and version control.

Course Outcomes

By the end of the course the students will be able to:

- Build web applications using Rails.
- Manipulate data using both imperative and functional programming techniques.
- Model real-world systems using object-oriented design.
- Write HTML & CSS to create elegant web pages.
- Use ActiveRecord to store and retrieve information from a database.

The students would have built fundamental first-principles based knowledge about web development and the practical chops to use them to build real-world software. They would also have learnt what it is to work in a professional software environment, helping build a strong foundation for their transition to the industry as competent professionals.

Prerequisites

Students should have completed *Web Development 101,* before beginning this course. Students should have access to a computer with a modern OS (Windows 10 or above, Ubuntu 20.04 and above, macOS 10.15 and above).

Overview

Through the course, the student will work up to build a Todo Management application using Rails, PostgreSQL, HTML, and CSS. The app will be hosted on the cloud using Heroku.

They will then independently build an online ordering and point-of-sale system similar to Swiggy as their capstone project. It will be a microcosm of a production web application and the challenges and trade-offs that come with it.

Being an industry-led course, the students will also be exposed to professional practices like code reviews, code quality, and version control (git). They'll have access to a Web Development Community where they are encouraged to ask well-crafted and specific questions, a valuable skill in a professional setting.

Course Content

Level 1 - Setting up your development environment

Intro to Git and Github

- GitHub: How To
- More about Git and collaboration

Level 2 - Introduction to Ruby

The essential Ruby programming workflow

- What exactly is a program
- Setting up a developer environment for Ruby
- Setting up VSCode for Ruby development
- Running our first program
- Using irb
- Getting comfortable with the *nix command line
- Using git to version control our code
- Writing meaningful git messages

Basics of the Ruby Programming Language

- Variables and data types
- Arrays, our first collections data structure
- Storing "key: value" lists in Ruby Hashes
- Functions: unit of abstraction and unit of naming
- Symbols & Blocks
- Let's write a program! (feat. recursive functions and HTML rendering)

Milestone Target - Write a recursive DNS resolver

Write a Ruby program which does domain name resolution and prints the lookup chain, until it resolves to an IPv4 address.

Level 3 - Object-oriented programming

Getting classy with objects!

- What is object-oriented programming and why do we need it?
- Defining a class
- Object-oriented design principle: Tell, don't ask
- Composition
- Inheritance

Milestone target - Implement a todo list using classes

Implementing Ruby command line todo list app using classes by following the naming conventions and formatting your code properly.

Level 4 - Databases and ActiveRecord

Introduction

- Why databases?
- Setting up PostgreSQL (Ubuntu & WSL)

ActiveRecord

- RubyGems: sharing and reusing code with the world
- Basics: connect to the database and create a table
- Creating ActiveRecord models and manipulating data

- Adding our own code to ActiveRecord classes

Milestone target - Build a Todo Management CLI app with Database

Level 5 - Backend Web development with Rails

Building a Ruby on Rails web application

- Hello world with Rails!
- Connect our PostgreSQL database to the Rails application
- Video Tutorials: Create a Rails project and connect it to the database!
- Add two features: create a new todo, and mark an existing todo as completed

Milestone target - Add user management to the Todo Manager app

Level 6 - Beautiful Websites with HTML and CSS

HTML and CSS

- Introduction
- Designing with CSS
- CSS classes, selectors, and specificity
- Layouts in CSS using display block and Flexbox

Milestone target - Recreate the Todo Manager design with HTML & CSS

Level 7 - ERB Templating and Asset Pipeline

Rendering dynamic HTML & CSS with ERB templating

- Render dynamic data inside HTML with ERB templates
- Use view templates for common elements like headers and footers
- What is MVC, and notes on the `public` folder
- The Asset Pipeline

Milestone target - Deploy your application to Heroku & Use ERB to render todo page and deploy to Heroku

Level 8 - HTML forms and Rails form helpers to save and accept user inputs

HTML Forms

- Introduction
- What is the <form> element and why do we need it?
- Adding a new Todo with the help of <form>
- Rails form helper for updating an existing todo
- Deleting a ToDo

Rails Authenticity Token verification and CSRF

- What is Cross Site Request Forgery (CSRF) and why should we care?
- Authenticity Token - how Rails helps us prevent CSRF attacks

Milestone target - Make the Todo page fully functional with forms

Level 9 - User Authentication and final wrap-up

ActiveRecord migrations and associations

- Introduction
- Create the `users` table with an ActiveRecord migration
- ActiveRecord association: belongs_to and has_many

Signing up new users

- Creating a user sign-up page
- Storing passwords with bcrypt and password digests

Cookies, sessions, and the user authentication workflow

- Create a simple sign-in page which verifies the user's password
- What exactly is a cookie and why should you care?
- Let us store the signed-in user's id in the session!
- Implement sign-out by resetting the user session
- A logged-in user should see and modify only their own todos and nobody else's

Flash messages and ActiveRecord validations

- Showing one-off messages to users with ActionDispatch::Flash
- Ensure no blank todos are ever created, with ActiveRecord validations

Milestone target - Complete and deploy the final Todo Manager SaaS application

Capstone Project

Build a Cafeteria Management Webapp, or a Build Your Own Capstone (BYOC) project.

Scheme of Continuous Evaluation

The pupil centered pedagogy features a scheme of continuous evaluation that enables faculty to give personalized feedback to each student across their lesson plans.

The WD course curriculum is organised into levels. Students start at Level 1, go through and complete the lessons within a level and move on to the next level. The lessons can be completed in one of several ways:

- **Marking as complete:** Students can go through the content and mark the lesson as complete.
- **Taking a quiz:** Some lessons have quizzes at the end that students can take to self-assess their understanding of the concepts covered in the lesson.
- **Visiting a link:** Some lessons have additional reading that students will be directed to after going through the content on the LMS.
- **Submitting work for review:** Some lessons (usually *milestone targets*) will require students to submit work for review. These submissions could be reviewed either automatically (using tests), or by coaches manually. Such submissions are graded - more on this below.

WD101

In important levels, there are **milestone targets,** where students are given a problem statement and instructions to complete the level. The students work on this task and submit it for review. The submissions go through a set of automated tests and if the submission passes the test, it is accepted, else it is rejected and the students are given feedback which will allow them to work on the same again and resubmit to get a passing grade.

The student needs to complete all milestone targets and get a passing grade in each of them to progress in the course.

At the end of the course, there is a *Capstone Project*. Once the students complete all the levels in the course, they work on the Capstone Project where they incorporate all the learnings acquired in the course towards building a project based on the specifications provided. The coach reviews the submission, and based on whether it meets the evaluation rubric, either accepts or rejects it (giving feedback).

WD201

In important levels, there are **milestone targets,** where students are given a problem statement and instructions to complete the level. The students work on this task and submit it for review by the coach. The coach goes through the work and assigns a grade and gives feedback to the student for improvement. The student can then work again on the task, based on the feedback, and resubmit the work for re-evaluation by the coach. This leads to better grades and better understanding of topics.

The student needs to complete milestone targets and receive a passing grade. This continuous assessment and feedback, and provision to resubmit the work is crucial for students to get better with the skills being taught in the course. The continuous assessment accounts for upto 50% of total marks ***(Internal Marks)*** in the course.

At the end of the course, there is a *Capstone Project*. Once the students complete all the levels in the course, they work on a Capstone Project where they incorporate all the learnings acquired in the course towards building a project based on the specifications provided.

They demonstrate the project to teaching assistants, coach(es), and / or industry-experts, and are assigned marks and given feedback. The Capstone Project accounts for the remaining 50% of total marks ***(External Marks)*** in the course.