Web Programming I

Lecture 01

Basic Course Information

Course Name	**Web Programming I**
ECTS	**7**
Duration	**16 Weeks**
Lecture	**32 Hours (2 hrs/week)**
Lab	**48 Hours (3 hrs/week)**
Tutorial	**16 Hours (1 hr/week)**
Home Study	**96 Hours (6 hrs/week)**

Course Objective

Learn how to develop modern web applications

Learning outcomes

- Explain the components involved in enterprise web application development
- Design web solutions using MVC design pattern
- Build modern database-backed web application
- Design and implement RESTful APIs
- Secure web applications and REST APIs
- Test web applications
- Deploy web application in different modern production environments

Concepts Covered

- HTTP request methods, HTTP response, Sessions and Cookies
- REST API, Routing, Templating, Securing Web Applications
- Data storage, Serving Files, JSON/XML processing
- Concurrency, Testing, Cloud and Container-Based Deployment

Environment Used

	Concurrency	Avg. latency	Req / sec	Transfer / sec
Laravel	1	3.87ms	261.48	1.27MB
Laravel	100	108.86ms	917.27	6.04MB
Go	1	325.72µs	7365.48	34.27MB
Go	100	11.63ms	19967.31	92.91MB
Go	200	37.68ms	22653.22	105.41MB

Go Programming Language

- Designed to be simple and efficient
- Focuses on programmer effectiveness and speed
- Provides features of functional programming
- Supports Concurrency
- Supports web-development in its standard library
- Suitable for large-scale web application development

Large-scale web development requirements

- **Scalability**
  - **Vertical** and **Horizontal** scalability
- **Modularity**
  - Powerful interface construct, functional programing support, microservice development
- **Maintainability**
  - Simple and readable syntax, code formatting and documentation support, testing support, modern packaging system
- **High Performance**
  - Concurrency support that allows multiple requests to be processed at the same time

Assessments

- Quizzes during lecture and lab time (10 %)
- Assignments (10 %)
- Project
  - First Round Evaluation (Formative) - to give you feedback (20 %)
  - Second Round Evaluation and Demo (Summative) (10 %)
- Mid and Final Exams (50 %)

Project Requirements

- Initial step
  - Form project group (Max 5 members)
  - Create trello board and git repository (gitlab/github/bitbucket) account
  - Specify project description and feature list on the trello board
  - **Due October 06, 2019**

Project Requirements

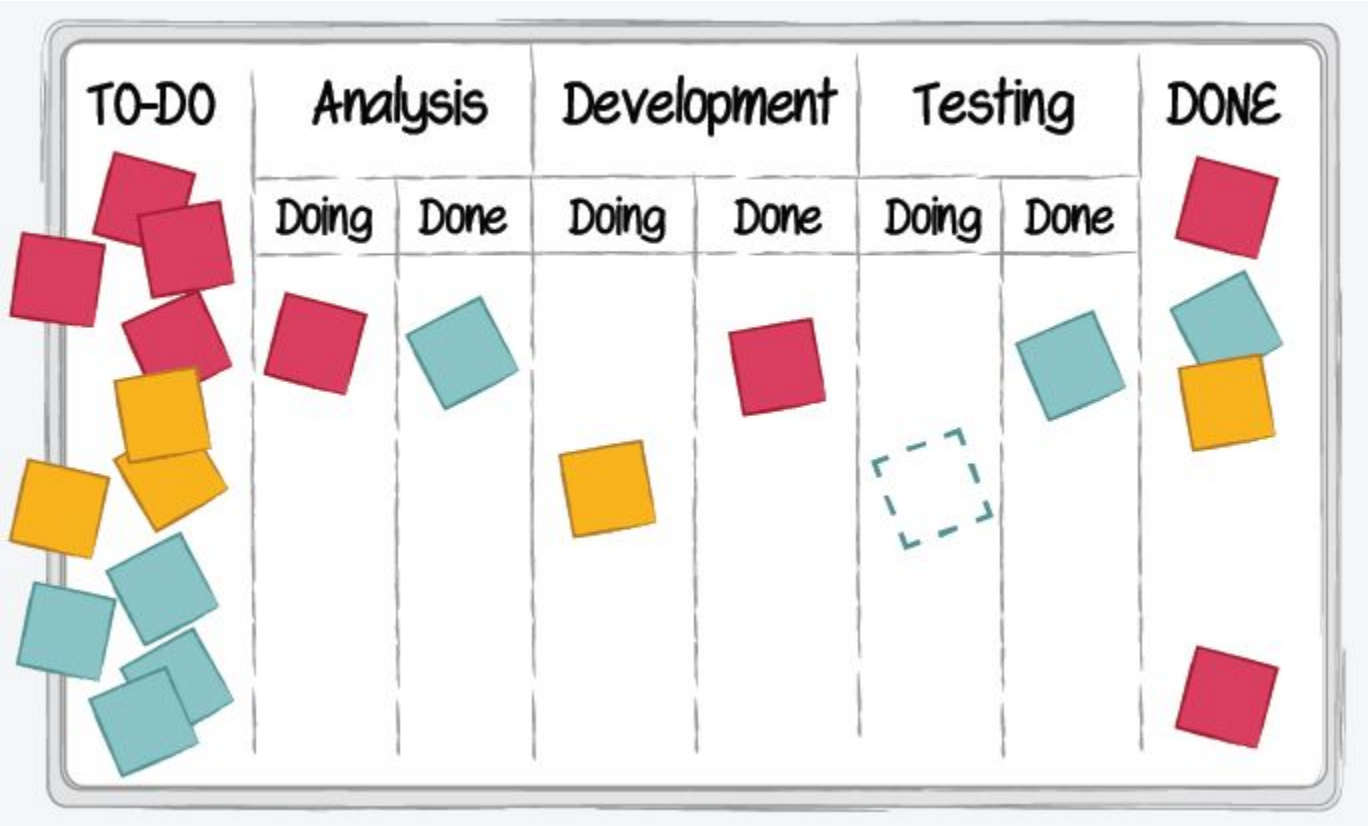
- On Trello Board provide
  - General project description (The problem the application will solve)
  - The Project title (use specific name related to the problem, do not use Web Programming I project or similar generic names)
  - Group members full name and Id number

Project Requirements

- Project progress is tracked using
  - Git repository commit history
  - Kanban board on Trello

Project Requirements

- Kanban board
  - You need to respect Kanban principles



Project Requirements

- Application feature documentation format
  - Should be written in the form of **User Story**
  - **As a < type of user/role >, I want < some goal > so that < some reason/benefit >**
  - **Acceptance Criteria**

Project Requirements

- User Story Example

Story Name: Customer Order

Description:

As a **Customer**, I need **to place an order** so that **I can have food delivered to my house**

Project Requirements

- User Story Example

As a **Customer**, I need **to place an order** so that **I can have food delivered to my house**

Acceptance Criteria:

Functional:

- Can I save my order and come back to it later?
- Can I change my order before I pay for it?
- Can I see a running total of the cost of what I have chosen so far?

Project Requirements

- User Story Example

As a **Customer**, I need **to place an order** so that **I can have food delivered to my house**

Acceptance Criteria:

Non-functional: Availability:

- Can I place an order at any time (24 hours per day or 24/7/365)?
- Can I view the order at any time (24 hours per day or 24/7/365) up to and including delivery?

Project Requirements

- User Story Example

As a **Customer**, I need **to place an order** so that **I can have food delivered to my house**

Acceptance Criteria:

Non-functional: Security:

- Are unauthorised persons and other customers prevented from viewing my order?

Project Requirements

- User Story Example

As a **Customer**, I need **to place an order** so that **I can have food delivered to my house**

Acceptance Criteria:

Non-functional: Security:

- Are unauthorised persons and other customers prevented from viewing my order?

Course Content

- **Week 2- Introducing Go**
  - Packages, variables, and functions
  - Flow control statements: for, if, if else, switch and defer
- **Week 3- Introducing Go**
  - Structs, slices, and maps
  - Methods and interfaces
  - Concurrency

Course Content

- Week 4 - Using Go for Web Application
- Week 5 - HTTP Request/Response
- Week 6 and 7- Templating and Routing
- Week 8 and 9 - Data persistency
- Week 10 - Web Services
- Week 11 - Securing Web Application
- Week 12 - Testing Web Application

Course Content

- **Week 13 - Concurrency for Performance Improvement**
- **Week 14 - Internationalization and Localization**
- **Week 15 - Middleware and Frameworks**
- **Week 16 - Deployment Options**

Resources

- For Go
  - A Tour of Go (https://tour.golang.org/)
  - The Go Programming Language (Addison-Wesley Professional Computing Series) 1st Edition, by Alan A. A. Donovan and Brian W. Kernighan
  - Effective Go (https://golang.org/doc/effective_go.html)
- For Web Development
  - Building Web Apps with Go (https://codegangsta.gitbooks.io/building-web-apps-with-go/content/)
  - Build Web Application with Golang (https://astaxie.gitbooks.io/build-web-application-with-golang/content/en/)