WESTMINSTER INTERNATIONAL UNIVERSITY IN TASHKENT
An Accredited Institution of the University of Westminster (UK)

Module name and code	Web Technologies 4BUIS011C
Coursework weighting	50%
Lecturer setting the task	Bunyod Khoshimkhujaev bkhoshimkhujaev@wiut.uz
Submission deadline	April 7, 2021 / 23:59
Results date and type of feedback	3 weeks after submission deadline (oral feedback)

The coursework checks the following learning outcomes

1. Illustrate how client-server architecture can be applied to build fully fledged dynamic web applications.
2. Define routes and logic/controllers to handle user requests to the web application.
3. Develop basic web applications from scratch as well as using modern web frameworks. Assess how frameworks can simplify the development process.
4. Deploy web applications using both hand-coding and appropriate software tools to the live server. Perform basic server configuration for properly serving user requests.

Task

This is individual coursework. You need to build fully fledged dynamic web applications chosen from the list below:

- Bug reporting application (users can submit bug report by opening a ticket and vendor can respond)
- Blog application (users can create posts, update or delete them, see list and detailed pages)
- To do application (users can create tasks, change their status, delete and review, etc.)
- HR application for storing employee data (users can add employees, change their status, etc.)
- Image gallery app (users can add images with description and see uploaded images)

A detailed description of the task

1. Define documentation and project structure - 15 marks

This part covers 2 aspects of the web application development:

- Basic documentation (i.e. README.md)
- Project structure definition

The description of the report is given below and you should follow its structure exactly as it is given

- Documentation - brief description of the application formatted in markdown (README.md file) Brief description must include (but not limited to) **(8 marks)**:
  - — about the app

— step by step instructions on how to run the app locally including dependencies' installation

— application dependencies' list

— user story or condensed down functionality description

— link to the public repo with web application source code on github

- project structure – files of the application is properly structured **(7 marks)**

i.e. expected structure of the project. It is strongly recommended to follow the very structure (if you prefer different structure please justify your choice in the README.md file):

/web-application-root	name of the folder usually matches the name of your project
app.js	the name of the file can be **[app.js]** or **[index.js]**
package.json	project configuration file. usually initiated with **[npm init]**
.gitignore	include /node_modules folder to this file. make sure not to upload the **[node_modules]** folder itself to github or Intranet.
/node_modules	this folder is required for dependencies. do not change the name of the folder
/public /images /javascripts /styles style.css	public folder is for keeping files essential for building user-interface (client-side)
/routes index.js users.js	route files will keep controllers/handlers for specific route groups. indicated route names are examples and a student must name his/her own route files depending on the context.
/views index.pug layout.pug	**[views]** name for the 'templates' folder is expected but not required. This can be changed but requires specific steps.

Important:

Do not upload [node_modules] folder to github or Intranet. Add [node_modules] to .gitignore file.

2. Develop web application - 85 marks

- The application should include these elements and functionality:
- Create / Read / Update / Delete operations with proper error handling and validation for user inputs. Usage of external libraries (i.e. for form validation) is encouraged. **(max 28 marks)**
- Proper route grouping (i.e. users, modules, tasks. etc.) **(max 7 marks)**
- The UI (front-end) of the web application must reflect the functionality of the application layer. **(max 20 marks)**
- Basic REST API which will send data (i.e. list of items) to the client in the JSON data format and corresponding page that will show the data to the user. **(max 10 marks)** API must follow these guidelines:
  - URL endpoint string must follow this pattern: '/api/v1/{ your_item_list }
  - REST API must respond with JSON formatted data
  - Requests to the defined URL endpoint must be done using native **fetch()** function

- Host your application's source code on Github (https://github.com - make public repo and share the link. You must create an account before pushing your code to the very platform) and make the related actions transparent. Your commits and pushes must show the development progress over the time. Commits must be properly formatted. Please refer to these articles to understand how to properly define commit messages:
    - http://karma-runner.github.io/0.13/dev/git-commit-msg.html
    - https://chris.beams.io/posts/git-commit/

  A few commits won't be evaluated. You must have at least 5 commits that will show your progress over the time **(10 marks)**

- Deploy your application online on Glitch platform (https://glitch.com/) and clearly indicate the URL of your web application in the project documentation. **(10 marks)**

Homepage of your web application must include the text "This web application was created to fulfill Web Technology module's requirements and does not represent an actual company or service". Check that the coursework has a standard cover page with all relevant data.

Submission requirements

An electronic version of your coursework must be submitted through the university Intranet system. To do so you need to follow these steps:

- Archive (use .zip format) your files and name it xxxxx.zip, where xxxxx is your ID number
- Go to the Web Technology module's coursework submission page.
- Attach appropriate files and submit.

The size of the archive must not exceed 15Mb. If you have media files (video, images, etc.) optimize or host them on available free platforms (i.e. youtube) and include only reference to the very resources. **Don't include project dependencies' folder (node_modules).**

General notes:

Please ensure that you work individually on this coursework. Plagiarism and close collaboration will not be tolerated and it will be considered an assessment offense. According to the Essential Information Handbook of Academic Regulations, any student may be invited for the oral viva (Please, see the regulations for full details).

Assessment criteria

Range	Description
70+	Everything is correct and clearly explained. A lot of work has gone into the development of the web site, proper use of web technologies (following the widely/suggested coding standards, clean code, accessible structure, etc.). Use of extra functions and features. Shows very advanced use of web programming languages with interactions between them.
60-69	Everything is correct but the description of the work done on the project is not very clear. Quite a lot of work on the development of the site. Good use of extra features and functionalities. Shows an advanced understanding of web programming languages.

50-59	A fair amount of work has gone into the development of the web application. An attempt has been made to add extra features and functionalities to the website. Shows an intermediate understanding of web programming languages.
40-49	Required sections are minimally covered. Adequate clarity and focus on requirements and objectives. Enough work has gone into the development of the site. The moderate use of web technologies. Shows an understanding of web programming languages.
30-39	Some work has gone into the development of the site. A mediocre attempt to use all web technologies.
0-29	Requirements are partially covered. Web application does not work.

Marking scheme

Define documentation and project structure	**15**
Documentation	8
Project structure definition	7
Web application	**85**
CRUD (create-read-update-delete) operations	28
Properly defined routing with appropriate URL naming	7
Properly designed UI/UX that corresponds to the back-end functionality	20
Properly developed basic REST API	10
Hosting the source code on github. Actions (commits, pushes, etc.) must be transparent/public for evaluation.	10
Hosting the website	10
Overall	**100**