CMPS 356 Enterprise Application Development - Spring 2018

**Lab 7 – Web APIs**

|  |  |
| --- | --- |
| **Student Name** |  |
| **Student Id** |  |
| **Email** |  |

**Grading Rubric - - In the *Functionality* column please specify either: *Working (completed x%)*, *Not Working (completed x%)* or *Not done*.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Criteria | % | Functionality\* | Quality of the implementation | Score |
| Part A - Extend the Banking App to asynchronously read/write data from the accounts.json file and make the App functionality accessible via Web API | 30 |  |  |  |
| 1. Get /api/accounts/ Returns all accounts 2. Get /api/accounts/:id Returns an account by id 3. Post /api/accounts Adds an account 4. Put /api/accounts/:id Updates an account 5. Delete /api/accounts/:id Deletes an account by id |  |  |  |
| Part B - Extend the GPA Calculator to asynchronously read/write data  from students.json file and make the App functionality accessible via  Web API | 70 |  |  |  |
| 1. Get => /api/students/ =>Returns all students 2. Get => /api/students/avgGPA Returns the average => GPA for all students 3. Get => /api/students/top2students Returns the top 2 students with highest GPA. 4. Get =>/api/students/:id Returns a student by student Id 5. Post => /api/students/:studentId/courses Adds a course to the student’s courses 6. Put=>/api/students/:studentId/courses/:courseCode => Updates a student course 7. Delete=>/api/students/:studentId/courses/:courseCode => Deletes a student course |  |  |  |  |
| Total | 100 |  |  |  |
| Copying and/or plagiarism or not being able to explain or answer questions about the implementation | -100 |  |  |  |

**\* Possible grading for functionality**: ***Complete and*** ***Working*** (get 70% of the assigned grade), ***Complete and*** ***Not*** ***working*** (lose 40% of assigned grade) and ***Not done*** get 0. The remaining grade is assigned to the quality of the implementation. In case your implementation is not working then 40% of the grade will be lost and the remaining 60% will be determined based on of the code quality and how close your solution to the working implementation. Quality includes meaningful naming of identifiers, no redundant code, simple and efficient design, clean code without unnecessary files/code, use of comments where necessary, proper white space and indentation. **Marks will be reduced** forcode duplication, poor/inefficient coding practices, poor naming of identifiers and unnecessary complex/poor user interface design.

## Testing evidence