

GitHub Link: <https://github.com/asantha-dilshan/Assignment-05-code.git>

Vercel Link: <https://assignment-05-code.vercel.app/>

## Server.js

```
1 // WEB700 - Assignment 05
2 * I declare that this assignment is my own work in accordance with Seneca Academic Policy.
3 * No part of this assignment has been copied manually or electronically from any other source
4 * (including 3rd party web sites) or distributed to other students.
5
6 * Name: Asantha Dilshan Jayasekara Misurumana Arachchige
7 * Student ID: 170388235
8 * Date: 03/25/2025
9 * Online (Vercel) link: https://assignment-05-code.vercel.app/
10
11
12
13 const express = require("express");
14 const path = require("path");
15 const collegeData = require("../modules/collegeData");
16 const app = express();
17 const helpers = require("../helpers");
18 const expressLayouts = require("express-ejs-layouts");
19
20 // Set EJS as the view engine
21 app.set("layout", "layouts/main");
22 app.set("views", path.join(__dirname, "views"));
23 app.set("view engine", "ejs");
24 app.use(expressLayouts);
25 app.use(express.static(path.join(__dirname, "public")));
26
27 app.use(function (req, res, next) {
28   let route = req.path.substring(1);
29   app.locals.activeRoute =
30     "/" +
31     (route.split("/")[1]
32       ? route.replace(/\/(!.)/, "")
33       : route.replace(/\/(!.)/, ""));
34   next();
35 });
36
37 app.use(express.urlencoded({ extended: true }));
38 const navLink = (url, options) => {
39   helpers.navLink(app.locals.activeRoute, url, options);
40 }
41
42 // Route to Serve addStudent.html
43 app.get("/students/add", (req, res) => {
44   res.sendFile(path.join(__dirname, "views", "addStudent.html"));
45 });
46
47 // Home Page Route
48 app.get("/home", (req, res) => {
49   res.render("home", { navlink: navLink });
50 });
```

```
46 // Home Page Route
47 app.get("/home", (req, res) => {
48   res.render("home", { navlink: navLink });
49 });
50
51 // / Route
52 app.get("/", (req, res) => {
53   res.render("home", { navlink: navLink });
54 });
55
56 // About Page Route
57 app.get("/about", (req, res) => {
58   res.render("about", { navlink: navLink });
59 });
60
61 // HTML Demo Page Route
62 app.get("/htmlDemo", (req, res) => {
63   res.render("htmlDemo", { navlink: navLink });
64 });
65
66 // Route for Add Student
67 app.get("/addStudents", (req, res) => {
68   res.render("addStudents", { navlink: navLink });
69 });
70
71 // POST /students/add
72 app.post("/students/add", (req, res) => {
73   collegeData
74     .addStudent(req.body)
75     .then(() => {
76       res.redirect("/students"); // Or res.render("student-added", { navlink: navLink });
77     })
78     .catch((err) => {
79       res.status(500).render("error", { message: "Error adding student: " + err, navlink: navLink });
80     });
81 });
82
83 // Get all students or by course
84 app.get("/students", (req, res) => {
85   if (req.query.course) {
86     collegeData
87       .getStudentsByCourse(req.query.course)
88       .then((students) => {
89         res.render("students", { students: students, navlink: navLink });
90       })
91       .catch(() => {
92         res.render("students", { message: "no results", navlink: navLink });
93       });
94   } else {
95     collegeData
96       .getStudents()
97       .then((students) => {
98         res.render("students", { students: students, navlink: navLink });
99       })
100      .catch(() => {
101        res.render("students", { message: "no results", navlink: navLink });
102      });
103   }
104 });
```

WEB700\_Assignment5\_Asintha Dilshan Jayasekara (170388235)

The image shows a screenshot of a Visual Studio Code editor window. The top bar displays the standard menu (File, Edit, Selection, View, Go, Run, Terminal, Help) and a search bar. The left sidebar contains the Explorer, Search, and Run and Debug views. The Explorer view shows a file tree with folders like 'main.js', 'server.js', and 'views'. The main editor area displays the 'server.js' file, which contains JavaScript code for a Node.js application. The code includes imports for 'express', 'mongoose', and 'dotenv', and defines routes for GET, POST, and PUT requests. The code also includes comments and error handling. The status bar at the bottom shows the file is named 'server.js' and is located in the 'main.js' folder. The bottom right corner shows the system tray with the date and time (3/26/2025, 4:15 AM) and the language (English, US).

[illegible]

WEB700\_Assignment5\_Asintha Dilshan Jayasekara (170388235)

The screenshot shows a VS Code editor with a Node.js project. The Explorer sidebar on the left lists the following files and folders:

- main.js
- server.js
- views
  - layouts
- main.js
- 404.html
- about.js
- addStudents.js
- course.js
- courses.js
- error.js
- home.js
- htmlDemo.js
- student.js
- students.js
- gignore
- helpers.js
- package-lock.json
- package.json
- server.js
- vercel.json

The main editor area displays the 'server.js' file, which contains the following code:

```

100 // ...
101 // res.send()
102 //
103 // <html>
104 // <head>
105 // <title>Student Added</title>
106 // <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css">
107 // </head>
108 // <body class="d-flex justify-content-center align-items-center vh-100">
109 // <div class="text-center">
110 //   <h2 class="mb-4">Student Added Successfully</h2>
111 //   <a href="/home" class="btn btn-primary">Go back to Home</a>
112 //   <a href="/students" class="btn btn-success">View All Students</a>
113 // </div>
114 // </body>
115 // </html>
116 // })
117 // .catch(err => res.status(500).send(err));
118 // });
119 //
120 // Handle 404 Errors
121 app.use((req, res) => {
122 |   res.status(404).sendFile(path.join(__dirname, "views", "404.html"));
123 | });
124 //
125 // async () => {
126 |   try {
127 |     await colledgeData.initialize(); then() => {
128 |       app.listen(process.env.PORT, () => {
129 |         console.log("Server listening on port ${process.env.PORT}");
130 |       });
131 |     };
132 |     console.log("Data initialized successfully");
133 |   } catch (err) {
134 |     console.error("Failed to initialize data: ", err);
135 |   }
136 | }());
137 //
138 // Export app for Vercel
139 module.exports = app;
140

```

The right sidebar shows a file explorer for the 'public' directory, listing files like 'index.html', 'styles.css', and 'scripts.js'.

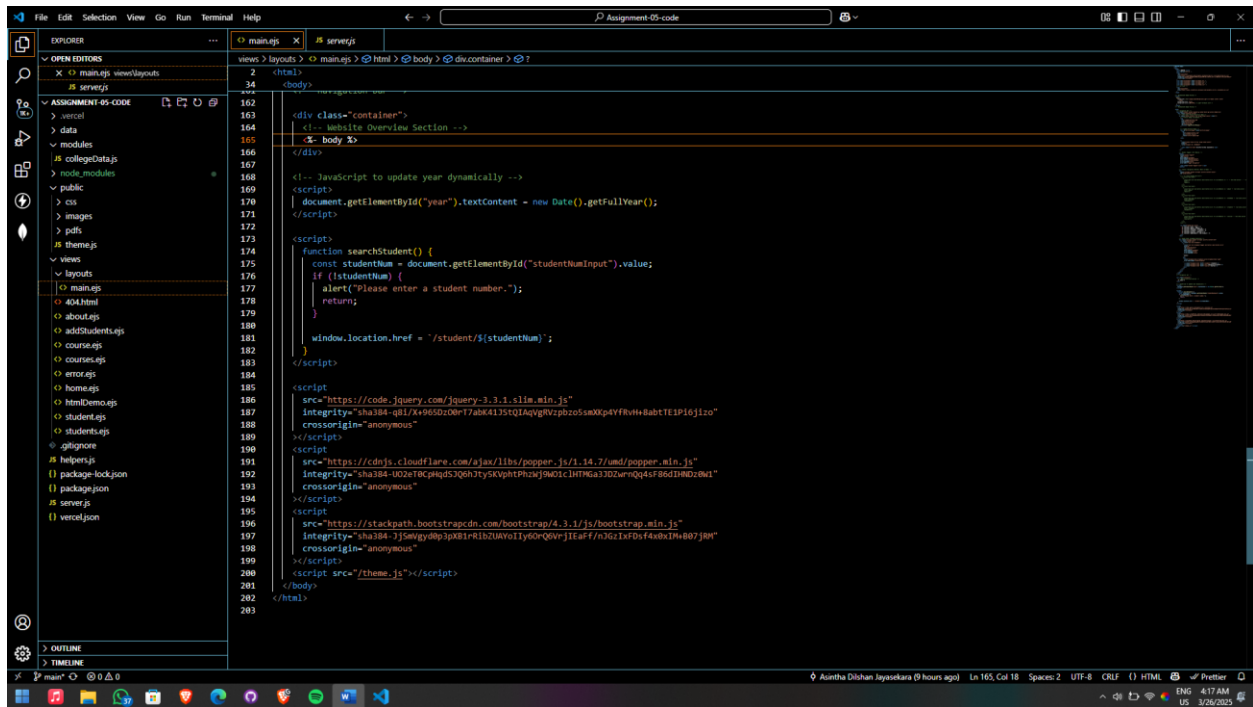
main.ejs

## WEB700\_Assignment5\_Asintha Dilshan Jayasekara (170388235)

```
2 <html>
3 <body>
4 <div class="nav-bar navbar-expand-lg navbar-dark bg-custom shadow-lg">
5 <div class="container-fluid">
6 <div class="navbar-brand d-flex align-items-center" href="#">
7 <!-- Hidden Profile Popup -->
8 <div id="profile-popup" class="profile-popup">
9 
14 </div>
15 <div class="navbar-brand d-flex align-items-center"
16   href="#"
17   title="Student ID: 170388235"
18 >
19 <span class="nav-name">Asintha Dilshan Jayasekara</span>
20 </div>
21 <!-- Nav-Bar Toggle (For Mobile) -->
22 <button
23   class="navbar-toggler"
24   type="button"
25   data-toggle="collapse"
26   data-target="#navbarContent"
27   aria-controls="navbarContent"
28   aria-expanded="false"
29   aria-label="toggle navigation"
30 >
31 <span class="navbar-toggler-icon"></span>
32 </button>
33 <!-- Center: Navigation Buttons (Next to Name) -->
34 <div
35   class="collapse navbar-collapse justify-content-center"
36   id="navbarContent"
37 >
38 <ul class="navbar-nav ms-3">
39 <li class="nav-item">
40 <a
41   class="nav-link nav-button small-button mx-2"
42   href="#"
43   home="/#"
44 >
45 </li>
46 </ul>
47 </div>
48 </div>
49 </div>
50 </body>
51 </html>
```

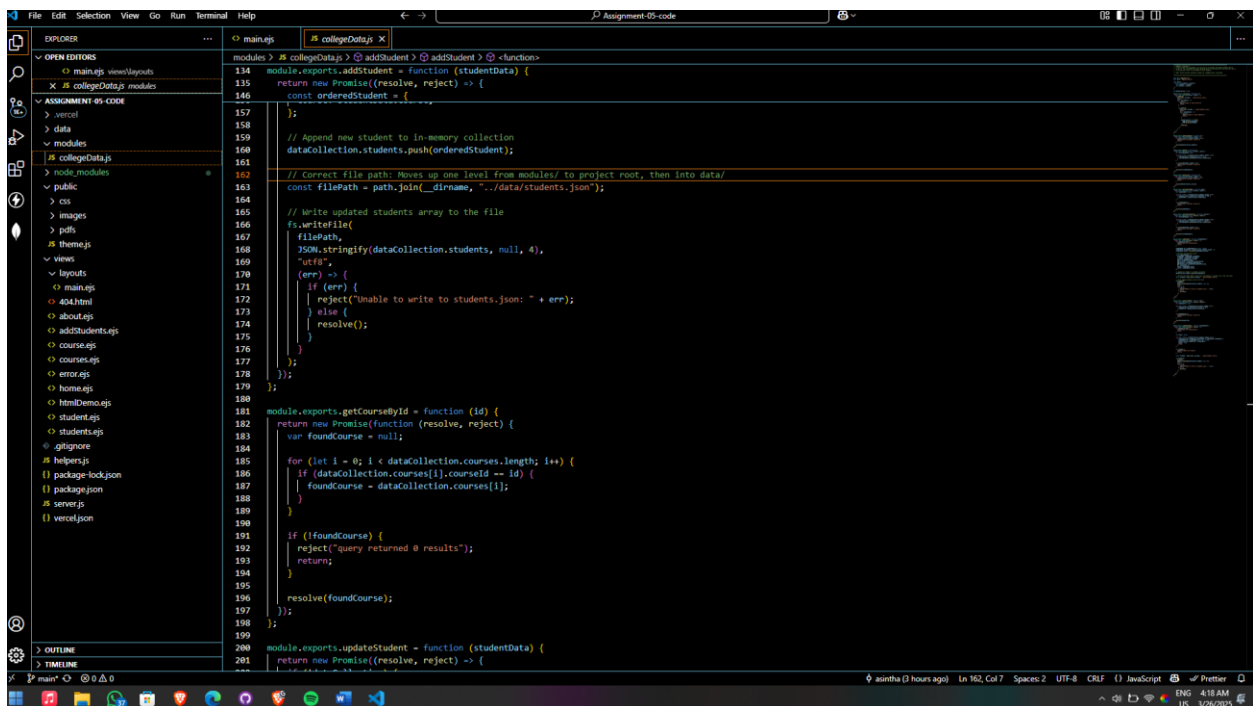
```
120 </li>
121 </ul>
122 </div>
123 <ul class="navbar-nav ms-3">
124 <li><a href="/#">Home</a></li>
125 <li><a href="/about/">About</a></li>
126 <li><a href="/htmlDemo/">HTML Demo</a></li>
127 <li><a href="/students/">Students</a></li>
128 <li><a href="/courses/">Courses</a></li>
129 <li><a href="/addStudents/">Add Student</a></li>
130 </ul>
131 </div>
132 </div>
133 <!-- Right Side: Dropdown Selection -->
134 <div class="collapse navbar-collapse justify-content-end">
135 <ul class="navbar-nav">
136 <li class="nav-item dropdown">
137 <a
138   class="nav-link dropdown-toggle nav-button small-button mx-2"
139   href="#"
140   id="navbarDropdown"
141   role="button"
142   data-toggle="dropdown"
143   aria-haspopup="true"
144   aria-expanded="false"
145 >
146 </a>
147 <div
148   class="dropdown-menu dropdown-custom dropdown-menu-right"
149   aria-labelledby="navbarDropdown"
150 >
151 <a class="dropdown-item" href="/students/">Students</a>
152 <a class="dropdown-item" href="/htmlDemo/">Teaching Assistants</a>
153 <a class="dropdown-item" href="/courses/">Courses</a>
154 </div>
155 </li>
156 </ul>
157 </div>
158 </div>
159 </div>
160 </div>
161 <!-- navigation bar -->
162 </div>
```

## WEB700\_Assignment5\_Asintha Dilshan Jayasekara (170388235)



```
114 <div class="container">
115   <!-- Website Overview Section -->
116   <div class="row">
117     <div class="col-md-6">
118       <h2>Website Overview</h2>
119     </div>
120     <div class="col-md-6">
121       <h3>About Us</h3>
122     </div>
123   </div>
124   <!-- JavaScript to update year dynamically -->
125   <script>
126     document.getElementById("year").textContent = new Date().getFullYear();
127   </script>
128   <script>
129     function searchStudent() {
130       const studentNum = document.getElementById("studentNumInput").value;
131       if (!studentNum) {
132         alert("Please enter a student number.");
133         return;
134       }
135       window.location.href = "/student/" + studentNum;
136     }
137   </script>
138   <script>
139     src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
140     integrity="sha384-q8i/X+965DzO0rT7abK41JstQIAqVgVz9pz8XMsXqMp49vHAQjyZ"
141     crossorigin="anonymous"
142   </script>
143   <script>
144     src="https://cdn.jsdelivr.net/npm/popper.js@1.14.7/umd/popper.min.js"
145     integrity="sha384-Uo2eTJKciBfUWmtLkCtfJ4oHspjNtS8oobc7J9XYL/V3G8FdeW87lR7YYH"
146     crossorigin="anonymous"
147   </script>
148   <script>
149     src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"
150     integrity="sha384-gdbQ9lFYGQHdH7e82Q07EP+86ZbRWK6FJ+d68hWNYN7H8BNfOZtsDD1N5d8L9CK"
151     crossorigin="anonymous"
152   </script>
153   <script src="theme.js"></script>
154 </div>
155 </body>
156 </html>
```

## collegeData.js



```
134 module.exports.addStudent = function (studentData) {
135   return new Promise((resolve, reject) => {
136     const orderedStudent = {
137       ...studentData,
138       id: Date.now().toString(36) + Math.random().toString(36).substr(2),
139     };
140     // Append new student to in-memory collection
141     dataCollection.students.push(orderedStudent);
142     // Correct file path: Moves up one level from modules/ to project root, then into data/
143     const filePath = path.join(__dirname, "../data/students.json");
144     // Write updated students array to the file
145     fs.writeFile(
146       filePath,
147       JSON.stringify(dataCollection.students, null, 4),
148       "utf8",
149       (err) => {
150         if (err) {
151           reject("Unable to write to students.json: " + err);
152         } else {
153           resolve();
154         }
155       }
156     );
157   });
158 };
159 module.exports.getCourseById = function (id) {
160   return new Promise(function (resolve, reject) {
161     var foundCourse = null;
162     for (let i = 0; i < dataCollection.courses.length; i++) {
163       if (dataCollection.courses[i].courseId === id) {
164         foundCourse = dataCollection.courses[i];
165       }
166     }
167     if (!foundCourse) {
168       reject("query returned 0 results");
169       return;
170     }
171     resolve(foundCourse);
172   });
173 };
174 module.exports.updateStudent = function (studentData) {
175   return new Promise((resolve, reject) => {
176     const { id, ...updateData } = studentData;
177     const index = dataCollection.students.findIndex(
178       (student) => student.id === id
179     );
180     if (index === -1) {
181       reject("Student not found");
182       return;
183     }
184     dataCollection.students[index] = {
185       ...dataCollection.students[index],
186       ...updateData,
187     };
188     // Write updated students array to the file
189     fs.writeFile(
190       filePath,
191       JSON.stringify(dataCollection.students, null, 4),
192       "utf8",
193       (err) => {
194         if (err) {
195           reject("Unable to write to students.json: " + err);
196         } else {
197           resolve();
198         }
199       }
200     );
201   });
202 };
203 module.exports.deleteStudent = function (id) {
204   return new Promise((resolve, reject) => {
205     const index = dataCollection.students.findIndex(
206       (student) => student.id === id
207     );
208     if (index === -1) {
209       reject("Student not found");
210       return;
211     }
212     dataCollection.students.splice(index, 1);
213     // Write updated students array to the file
214     fs.writeFile(
215       filePath,
216       JSON.stringify(dataCollection.students, null, 4),
217       "utf8",
218       (err) => {
219         if (err) {
220           reject("Unable to write to students.json: " + err);
221         } else {
222           resolve();
223         }
224       }
225     );
226   });
227 };
228 module.exports.getStudents = function () {
229   return new Promise((resolve, reject) => {
230     resolve(dataCollection.students);
231   });
232 };
233 module.exports.getStudentsByCourse = function (courseId) {
234   return new Promise((resolve, reject) => {
235     const students = dataCollection.students.filter(
236       (student) => student.courseId === courseId
237     );
238     resolve(students);
239   });
240 };
241 module.exports.getStudentsByYear = function (year) {
242   return new Promise((resolve, reject) => {
243     const students = dataCollection.students.filter(
244       (student) => student.year === year
245     );
246     resolve(students);
247   });
248 };
249 module.exports.getStudentsByLevel = function (level) {
250   return new Promise((resolve, reject) => {
251     const students = dataCollection.students.filter(
252       (student) => student.level === level
253     );
254     resolve(students);
255   });
256 };
257 module.exports.getStudentsByGender = function (gender) {
258   return new Promise((resolve, reject) => {
259     const students = dataCollection.students.filter(
260       (student) => student.gender === gender
261     );
262     resolve(students);
263   });
264 };
265 module.exports.getStudentsByAge = function (age) {
266   return new Promise((resolve, reject) => {
267     const students = dataCollection.students.filter(
268       (student) => student.age === age
269     );
270     resolve(students);
271   });
272 };
273 module.exports.getStudentsByAddress = function (address) {
274   return new Promise((resolve, reject) => {
275     const students = dataCollection.students.filter(
276       (student) => student.address === address
277     );
278     resolve(students);
279   });
280 };
281 module.exports.getStudentsByPhone = function (phone) {
282   return new Promise((resolve, reject) => {
283     const students = dataCollection.students.filter(
284       (student) => student.phone === phone
285     );
286     resolve(students);
287   });
288 };
289 module.exports.getStudentsByEmail = function (email) {
290   return new Promise((resolve, reject) => {
291     const students = dataCollection.students.filter(
292       (student) => student.email === email
293     );
294     resolve(students);
295   });
296 };
297 module.exports.getStudentsByName = function (name) {
298   return new Promise((resolve, reject) => {
299     const students = dataCollection.students.filter(
300       (student) => student.name === name
301     );
302     resolve(students);
303   });
304 };
305 module.exports.getStudentsByDateOfBirth = function (dateOfBirth) {
306   return new Promise((resolve, reject) => {
307     const students = dataCollection.students.filter(
308       (student) => student.dateOfBirth === dateOfBirth
309     );
310     resolve(students);
311   });
312 };
313 module.exports.getStudentsByNationality = function (nationality) {
314   return new Promise((resolve, reject) => {
315     const students = dataCollection.students.filter(
316       (student) => student.nationality === nationality
317     );
318     resolve(students);
319   });
320 };
321 module.exports.getStudentsByReligion = function (religion) {
322   return new Promise((resolve, reject) => {
323     const students = dataCollection.students.filter(
324       (student) => student.religion === religion
325     );
326     resolve(students);
327   });
328 };
329 module.exports.getStudentsByMaritalStatus = function (maritalStatus) {
330   return new Promise((resolve, reject) => {
331     const students = dataCollection.students.filter(
332       (student) => student.maritalStatus === maritalStatus
333     );
334     resolve(students);
335   });
336 };
337 module.exports.getStudentsByOccupation = function (occupation) {
338   return new Promise((resolve, reject) => {
339     const students = dataCollection.students.filter(
340       (student) => student.occupation === occupation
341     );
342     resolve(students);
343   });
344 };
345 module.exports.getStudentsByIncome = function (income) {
346   return new Promise((resolve, reject) => {
347     const students = dataCollection.students.filter(
348       (student) => student.income === income
349     );
350     resolve(students);
351   });
352 };
353 module.exports.getStudentsByEducationLevel = function (educationLevel) {
354   return new Promise((resolve, reject) => {
355     const students = dataCollection.students.filter(
356       (student) => student.educationLevel === educationLevel
357     );
358     resolve(students);
359   });
360 };
361 module.exports.getStudentsByEmploymentStatus = function (employmentStatus) {
362   return new Promise((resolve, reject) => {
363     const students = dataCollection.students.filter(
364       (student) => student.employmentStatus === employmentStatus
365     );
366     resolve(students);
367   });
368 };
369 module.exports.getStudentsByHealthStatus = function (healthStatus) {
370   return new Promise((resolve, reject) => {
371     const students = dataCollection.students.filter(
372       (student) => student.healthStatus === healthStatus
373     );
374     resolve(students);
375   });
376 };
377 module.exports.getStudentsByMentalHealthStatus = function (mentalHealthStatus) {
378   return new Promise((resolve, reject) => {
379     const students = dataCollection.students.filter(
380       (student) => student.mentalHealthStatus === mentalHealthStatus
381     );
382     resolve(students);
383   });
384 };
385 module.exports.getStudentsByPhysicalHealthStatus = function (physicalHealthStatus) {
386   return new Promise((resolve, reject) => {
387     const students = dataCollection.students.filter(
388       (student) => student.physicalHealthStatus === physicalHealthStatus
389     );
390     resolve(students);
391   });
392 };
393 module.exports.getStudentsBySocialHealthStatus = function (socialHealthStatus) {
394   return new Promise((resolve, reject) => {
395     const students = dataCollection.students.filter(
396       (student) => student.socialHealthStatus === socialHealthStatus
397     );
398     resolve(students);
399   });
400 };
401 module.exports.getStudentsByEmotionalHealthStatus = function (emotionalHealthStatus) {
402   return new Promise((resolve, reject) => {
403     const students = dataCollection.students.filter(
404       (student) => student.emotionalHealthStatus === emotionalHealthStatus
405     );
406     resolve(students);
407   });
408 };
409 module.exports.getStudentsBySpiritualHealthStatus = function (spiritualHealthStatus) {
410   return new Promise((resolve, reject) => {
411     const students = dataCollection.students.filter(
412       (student) => student.spiritualHealthStatus === spiritualHealthStatus
413     );
414     resolve(students);
415   });
416 };
417 module.exports.getStudentsByIntellectualHealthStatus = function (intellectualHealthStatus) {
418   return new Promise((resolve, reject) => {
419     const students = dataCollection.students.filter(
420       (student) => student.intellectualHealthStatus === intellectualHealthStatus
421     );
422     resolve(students);
423   });
424 };
425 module.exports.getStudentsByMoralHealthStatus = function (moralHealthStatus) {
426   return new Promise((resolve, reject) => {
427     const students = dataCollection.students.filter(
428       (student) => student.moralHealthStatus === moralHealthStatus
429     );
430     resolve(students);
431   });
432 };
433 module.exports.getStudentsByPhysicalAppearance = function (physicalAppearance) {
434   return new Promise((resolve, reject) => {
435     const students = dataCollection.students.filter(
436       (student) => student.physicalAppearance === physicalAppearance
437     );
438     resolve(students);
439   });
440 };
441 module.exports.getStudentsByFacialFeatures = function (facialFeatures) {
442   return new Promise((resolve, reject) => {
443     const students = dataCollection.students.filter(
444       (student) => student.facialFeatures === facialFeatures
445     );
446     resolve(students);
447   });
448 };
449 module.exports.getStudentsByBodyFeatures = function (bodyFeatures) {
450   return new Promise((resolve, reject) => {
451     const students = dataCollection.students.filter(
452       (student) => student.bodyFeatures === bodyFeatures
453     );
454     resolve(students);
455   });
456 };
457 module.exports.getStudentsByVoiceFeatures = function (voiceFeatures) {
458   return new Promise((resolve, reject) => {
459     const students = dataCollection.students.filter(
460       (student) => student.voiceFeatures === voiceFeatures
461     );
462     resolve(students);
463   });
464 };
465 module.exports.getStudentsByScentFeatures = function (scentFeatures) {
466   return new Promise((resolve, reject) => {
467     const students = dataCollection.students.filter(
468       (student) => student.scentFeatures === scentFeatures
469     );
470     resolve(students);
471   });
472 };
473 module.exports.getStudentsByTasteFeatures = function (tasteFeatures) {
474   return new Promise((resolve, reject) => {
475     const students = dataCollection.students.filter(
476       (student) => student.tasteFeatures === tasteFeatures
477     );
478     resolve(students);
479   });
480 };
481 module.exports.getStudentsByTouchFeatures = function (touchFeatures) {
482   return new Promise((resolve, reject) => {
483     const students = dataCollection.students.filter(
484       (student) => student.touchFeatures === touchFeatures
485     );
486     resolve(students);
487   });
488 };
489 module.exports.getStudentsByTemperatureFeatures = function (temperatureFeatures) {
490   return new Promise((resolve, reject) => {
491     const students = dataCollection.students.filter(
492       (student) => student.temperatureFeatures === temperatureFeatures
493     );
494     resolve(students);
495   });
496 };
497 module.exports.getStudentsByHumidityFeatures = function (humidityFeatures) {
498   return new Promise((resolve, reject) => {
499     const students = dataCollection.students.filter(
500       (student) => student.humidityFeatures === humidityFeatures
501     );
502     resolve(students);
503   });
504 };
505 module.exports.getStudentsByAirPressureFeatures = function (airPressureFeatures) {
506   return new Promise((resolve, reject) => {
507     const students = dataCollection.students.filter(
508       (student) => student.airPressureFeatures === airPressureFeatures
509     );
510     resolve(students);
511   });
512 };
513 module.exports.getStudentsByLightFeatures = function (lightFeatures) {
514   return new Promise((resolve, reject) => {
515     const students = dataCollection.students.filter(
516       (student) => student.lightFeatures === lightFeatures
517     );
518     resolve(students);
519   });
520 };
521 module.exports.getStudentsBySoundFeatures = function (soundFeatures) {
522   return new Promise((resolve, reject) => {
523     const students = dataCollection.students.filter(
524       (student) => student.soundFeatures === soundFeatures
525     );
526     resolve(students);
527   });
528 };
529 module.exports.getStudentsBySmellFeatures = function (smellFeatures) {
530   return new Promise((resolve, reject) => {
531     const students = dataCollection.students.filter(
532       (student) => student.smellFeatures === smellFeatures
533     );
534     resolve(students);
535   });
536 };
537 module.exports.getStudentsByTasteFeatures = function (tasteFeatures) {
538   return new Promise((resolve, reject) => {
539     const students = dataCollection.students.filter(
540       (student) => student.tasteFeatures === tasteFeatures
541     );
542     resolve(students);
543   });
544 };
545 module.exports.getStudentsByTouchFeatures = function (touchFeatures) {
546   return new Promise((resolve, reject) => {
547     const students = dataCollection.students.filter(
548       (student) => student.touchFeatures === touchFeatures
549     );
550     resolve(students);
551   });
552 };
553 module.exports.getStudentsByTemperatureFeatures = function (temperatureFeatures) {
554   return new Promise((resolve, reject) => {
555     const students = dataCollection.students.filter(
556       (student) => student.temperatureFeatures === temperatureFeatures
557     );
558     resolve(students);
559   });
560 };
561 module.exports.getStudentsByHumidityFeatures = function (humidityFeatures) {
562   return new Promise((resolve, reject) => {
563     const students = dataCollection.students.filter(
564       (student) => student.humidityFeatures === humidityFeatures
565     );
566     resolve(students);
567   });
568 };
569 module.exports.getStudentsByAirPressureFeatures = function (airPressureFeatures) {
570   return new Promise((resolve, reject) => {
571     const students = dataCollection.students.filter(
572       (student) => student.airPressureFeatures === airPressureFeatures
573     );
574     resolve(students);
575   });
576 };
577 module.exports.getStudentsByLightFeatures = function (lightFeatures) {
578   return new Promise((resolve, reject) => {
579     const students = dataCollection.students.filter(
580       (student) => student.lightFeatures === lightFeatures
581     );
582     resolve(students);
583   });
584 };
585 module.exports.getStudentsBySoundFeatures = function (soundFeatures) {
586   return new Promise((resolve, reject) => {
587     const students = dataCollection.students.filter(
588       (student) => student.soundFeatures === soundFeatures
589     );
590     resolve(students);
591   });
592 };
593 module.exports.getStudentsBySmellFeatures = function (smellFeatures) {
594   return new Promise((resolve, reject) => {
595     const students = dataCollection.students.filter(
596       (student) => student.smellFeatures === smellFeatures
597     );
598     resolve(students);
599   });
600 };
601 module.exports.getStudentsByTasteFeatures = function (tasteFeatures) {
602   return new Promise((resolve, reject) => {
603     const students = dataCollection.students.filter(
604       (student) => student.tasteFeatures === tasteFeatures
605     );
606     resolve(students);
607   });
608 };
609 module.exports.getStudentsByTouchFeatures = function (touchFeatures) {
610   return new Promise((resolve, reject) => {
611     const students = dataCollection.students.filter(
612       (student) => student.touchFeatures === touchFeatures
613     );
614     resolve(students);
615   });
616 };
617 module.exports.getStudentsByTemperatureFeatures = function (temperatureFeatures) {
618   return new Promise((resolve, reject) => {
619     const students = dataCollection.students.filter(
620       (student) => student.temperatureFeatures === temperatureFeatures
621     );
622     resolve(students);
623   });
624 };
625 module.exports.getStudentsByHumidityFeatures = function (humidityFeatures) {
626   return new Promise((resolve, reject) => {
627     const students = dataCollection.students.filter(
628       (student) => student.humidityFeatures === humidityFeatures
629     );
630     resolve(students);
631   });
632 };
633 module.exports.getStudentsByAirPressureFeatures = function (airPressureFeatures) {
634   return new Promise((resolve, reject) => {
635     const students = dataCollection.students.filter(
636       (student) => student.airPressureFeatures === airPressureFeatures
637     );
638     resolve(students);
639   });
640 };
641 module.exports.getStudentsByLightFeatures = function (lightFeatures) {
642   return new Promise((resolve, reject) => {
643     const students = dataCollection.students.filter(
644       (student) => student.lightFeatures === lightFeatures
645     );
646     resolve(students);
647   });
648 };
649 module.exports.getStudentsBySoundFeatures = function (soundFeatures) {
650   return new Promise((resolve, reject) => {
651     const students = dataCollection.students.filter(
652       (student) => student.soundFeatures === soundFeatures
653     );
654     resolve(students);
655   });
656 };
657 module.exports.getStudentsBySmellFeatures = function (smellFeatures) {
658   return new Promise((resolve, reject) => {
659     const students = dataCollection.students.filter(
660       (student) => student.smellFeatures === smellFeatures
661     );
662     resolve(students);
663   });
664 };
665 module.exports.getStudentsByTasteFeatures = function (tasteFeatures) {
666   return new Promise((resolve, reject) => {
667     const students = dataCollection.students.filter(
668       (student) => student.tasteFeatures === tasteFeatures
669     );
670     resolve(students);
671   });
672 };
673 module.exports.getStudentsByTouchFeatures = function (touchFeatures) {
674   return new Promise((resolve, reject) => {
675     const students = dataCollection.students.filter(
676       (student) => student.touchFeatures === touchFeatures
677     );
678     resolve(students);
679   });
680 };
681 module.exports.getStudentsByTemperatureFeatures = function (temperatureFeatures) {
682   return new Promise((resolve, reject) => {
683     const students = dataCollection.students.filter(
684       (student) => student.temperatureFeatures === temperatureFeatures
685     );
686     resolve(students);
687   });
688 };
689 module.exports.getStudentsByHumidityFeatures = function (humidityFeatures) {
690   return new Promise((resolve, reject) => {
691     const students = dataCollection.students.filter(
692       (student) => student.humidityFeatures === humidityFeatures
693     );
694     resolve(students);
695   });
696 };
697 module.exports.getStudentsByAirPressureFeatures = function (airPressureFeatures) {
698   return new Promise((resolve, reject) => {
699     const students = dataCollection.students.filter(
700       (student) => student.airPressureFeatures === airPressureFeatures
701     );
702     resolve(students);
703   });
704 };
705 module.exports.getStudentsByLightFeatures = function (lightFeatures) {
706   return new Promise((resolve, reject) => {
707     const students = dataCollection.students.filter(
708       (student) => student.lightFeatures === lightFeatures
709     );
710     resolve(students);
711   });
712 };
713 module.exports.getStudentsBySoundFeatures = function (soundFeatures) {
714   return new Promise((resolve, reject) => {
715     const students = dataCollection.students.filter(
716       (student) => student.soundFeatures === soundFeatures
717     );
718     resolve(students);
719   });
720 };
721 module.exports.getStudentsBySmellFeatures = function (smellFeatures) {
722   return new Promise((resolve, reject) => {
723     const students = dataCollection.students.filter(
724       (student) => student.smellFeatures === smellFeatures
725     );
726     resolve(students);
727   });
728 };
729 module.exports.getStudentsByTasteFeatures = function (tasteFeatures) {
730   return new Promise((resolve, reject) => {
731     const students = dataCollection.students.filter(
732       (student) => student.tasteFeatures === tasteFeatures
733     );
734     resolve(students);
735   });
736 };
737 module.exports.getStudentsByTouchFeatures = function (touchFeatures) {
738   return new Promise((resolve, reject) => {
739     const students = dataCollection.students.filter(
740       (student) => student.touchFeatures === touchFeatures
741     );
742     resolve(students);
743   });
744 };
745 module.exports.getStudentsByTemperatureFeatures = function (temperatureFeatures) {
746   return new Promise((resolve, reject) => {
747     const students = dataCollection.students.filter(
748       (student) => student.temperatureFeatures === temperatureFeatures
749     );
750     resolve(students);
751   });
752 };
753 module.exports.getStudentsByHumidityFeatures = function (humidityFeatures) {
754   return new Promise((resolve, reject) => {
755     const students = dataCollection.students.filter(
756       (student) => student.humidityFeatures === humidityFeatures
757     );
758     resolve(students);
759   });
760 };
761 module.exports.getStudentsByAirPressureFeatures = function (airPressureFeatures) {
762   return new Promise((resolve, reject) => {
763     const students = dataCollection.students.filter(
764       (student) => student.airPressureFeatures === airPressureFeatures
765     );
766     resolve(students);
767   });
768 };
769 module.exports.getStudentsByLightFeatures = function (lightFeatures) {
770   return new Promise((resolve, reject) => {
771     const students = dataCollection.students.filter(
772       (student) => student.lightFeatures === lightFeatures
773     );
774     resolve(students);
775   });
776 };
777 module.exports.getStudentsBySoundFeatures = function (soundFeatures) {
778   return new Promise((resolve, reject) => {
779     const students = dataCollection.students.filter(
780       (student) => student.soundFeatures === soundFeatures
781     );
782     resolve(students);
783   });
784 };
785 module.exports.getStudentsBySmellFeatures = function (smellFeatures) {
786   return new Promise((resolve, reject) => {
787     const students = dataCollection.students.filter(
788       (student) => student.smellFeatures === smellFeatures
789     );
790     resolve(students);
791   });
792 };
793 module.exports.getStudentsByTasteFeatures = function (tasteFeatures) {
794   return new Promise((resolve, reject) => {
795     const students = dataCollection.students.filter(
796       (student) => student.tasteFeatures === tasteFeatures
797     );
798     resolve(students);
799   });
800 };
801 module.exports.getStudentsByTouchFeatures = function (touchFeatures) {
802   return new Promise((resolve, reject) => {
803     const students = dataCollection.students.filter(
804       (student) => student.touchFeatures === touchFeatures
805     );
806     resolve(students);
807   });
808 };
809 module.exports.getStudentsByTemperatureFeatures = function (temperatureFeatures) {
810   return new Promise((resolve, reject) => {
811     const students = dataCollection.students.filter(
812       (student) => student.temperatureFeatures === temperatureFeatures
813     );
814     resolve(students);
815   });
816 };
817 module.exports.getStudentsByHumidityFeatures = function (humidityFeatures) {
818   return new Promise((resolve, reject) => {
819     const students = dataCollection.students.filter(
820       (student) => student.humidityFeatures === humidityFeatures
821     );
822     resolve(students);
823   });
824 };
825 module.exports.getStudentsByAirPressureFeatures = function (airPressureFeatures) {
826   return new Promise((resolve, reject) => {
827     const students = dataCollection.students.filter(
828       (student) => student.airPressureFeatures === airPressureFeatures
829     );
830     resolve(students);
831   });
832 };
833 module.exports.getStudentsByLightFeatures = function (lightFeatures) {
834   return new Promise((resolve, reject) => {
835     const students = dataCollection.students.filter(
836       (student) => student.lightFeatures === lightFeatures
837     );
838     resolve(students);
839   });
840 };
841 module.exports.getStudentsBySoundFeatures = function (soundFeatures) {
842   return new Promise((resolve, reject) => {
843     const students = dataCollection.students.filter(
844       (student) => student.soundFeatures === soundFeatures
845     );
846     resolve(students);
847   });
848 };
849 module.exports.getStudentsBySmellFeatures = function (smellFeatures) {
850   return new Promise((resolve, reject) => {
851     const students = dataCollection.students.filter(
852       (student) => student.smellFeatures === smellFeatures
853     );
854     resolve(students);
855   });
856 };
857 module.exports.getStudentsByTasteFeatures = function (tasteFeatures) {
858   return new Promise((resolve, reject) => {
859     const students = dataCollection.students.filter(
860       (student) => student.tasteFeatures === tasteFeatures
861     );
862     resolve(students);
863   });
864 };
865 module.exports.getStudentsByTouchFeatures = function (touchFeatures) {
866   return new Promise((resolve, reject) => {
867     const students = dataCollection.students.filter(
868       (student) => student.touchFeatures === touchFeatures
869     );
870     resolve(students);
871   });
872 };
873 module.exports.getStudentsByTemperatureFeatures = function (temperatureFeatures) {
874   return new Promise((resolve, reject) => {
875     const students = dataCollection.students.filter(
876       (student) => student.temperatureFeatures === temperatureFeatures
877     );
878     resolve(students);
879   });
880 };
881 module.exports.getStudentsByHumidityFeatures = function (humidityFeatures) {
882   return new Promise((resolve, reject) => {
883     const students = dataCollection.students.filter(
884       (student) => student.humidityFeatures === humidityFeatures
885     );
886     resolve(students);
887   });
888 };
889 module.exports.getStudentsByAirPressureFeatures = function (airPressureFeatures) {
890   return new Promise((resolve, reject) => {
891     const students = dataCollection.students.filter(
892       (student) => student.airPressureFeatures === airPressureFeatures
893     );
894     resolve(students);
895   });
896 };
897 module.exports.getStudentsByLightFeatures = function (lightFeatures) {
898   return new Promise((resolve, reject) => {
899     const students = dataCollection.students.filter(
900       (student) => student.lightFeatures === lightFeatures
901     );
902     resolve(students);
903   });
904 };
905 module.exports.getStudentsBySoundFeatures = function (soundFeatures) {
906   return new Promise((resolve, reject) => {
907     const students = dataCollection.students.filter(
908       (student) => student.soundFeatures === soundFeatures
909     );
910     resolve(students);
911   });
912 };
913 module.exports.getStudentsBySmellFeatures = function (smellFeatures) {
914   return new Promise((resolve, reject) => {
915     const students = dataCollection.students.filter(
916       (student) => student.smellFeatures === smellFeatures
917     );
918     resolve(students);
919   });
920 };
921 module.exports.getStudentsByTasteFeatures = function (tasteFeatures) {
922   return new Promise((resolve, reject) => {
923     const students = dataCollection.students.filter(
924       (student) => student.tasteFeatures === tasteFeatures
925     );
926     resolve(students);
927   });
928 };
929 module.exports.getStudentsByTouchFeatures = function (touchFeatures) {
930   return new Promise((resolve, reject) => {
931     const students = dataCollection.students.filter(
932       (student) => student.touchFeatures === touchFeatures
933     );
934     resolve(students);
935   });
936 };
937 module.exports.getStudentsByTemperatureFeatures = function (temperatureFeatures) {
938   return new Promise((resolve, reject) => {
939     const students = dataCollection.students.filter(
940       (student) => student.temperatureFeatures === temperatureFeatures
941     );
942     resolve(students);
943   });
944 };
945 module.exports.getStudentsByHumidityFeatures = function (humidityFeatures) {
946   return new Promise((resolve, reject) => {
947     const students = dataCollection.students.filter(
948       (student) => student.humidityFeatures === humidityFeatures
949     );
950     resolve(students);
951   });
952 };
953 module.exports.getStudentsByAirPressureFeatures = function (airPressureFeatures) {
954   return new Promise((resolve, reject) => {
955     const students = dataCollection.students.filter(
956       (student) => student.airPressureFeatures === airPressureFeatures
957     );
958     resolve(students);
959   });
960 };
961 module.exports.getStudentsByLightFeatures = function (lightFeatures) {
962   return new Promise((resolve, reject) => {
963     const students = dataCollection.students.filter(
964       (student) => student.lightFeatures === lightFeatures
965     );
966     resolve(students);
967   });
968 };
969 module.exports.getStudentsBySoundFeatures = function (soundFeatures) {
970   return new Promise((resolve, reject) => {
971     const students = dataCollection.students.filter(
972       (student) => student.soundFeatures === soundFeatures
973     );
974     resolve(students);
975   });
976 };
977 module.exports.getStudentsBySmellFeatures = function (smellFeatures) {
978   return new Promise((resolve, reject) => {
979     const students = dataCollection.students.filter(
980       (student) => student.smellFeatures === smellFeatures
981     );
982     resolve(students);
983   });
984 };
985 module.exports.getStudentsByTasteFeatures = function (tasteFeatures) {
986   return new Promise((resolve, reject) => {
987     const students = dataCollection.students.filter(
988       (student) => student.tasteFeatures === tasteFeatures
989     );
990     resolve(students);
991   });
992 };
993 module.exports.getStudentsByTouchFeatures = function (touchFeatures) {
994   return new Promise((resolve, reject) => {
995     const students = dataCollection.students.filter(
996       (student) => student.touchFeatures === touchFeatures
997     );
998     resolve(students);
999   });
1000 };
1001 module.exports.getStudentsByTemperatureFeatures = function (temperatureFeatures) {
1002   return new Promise((resolve, reject) => {
1003     const students = dataCollection.students.filter(
1004       (student) => student.temperatureFeatures === temperatureFeatures
1005     );
1006     resolve(students);
1007   });
1008 };
1009 module.exports.getStudentsByHumidityFeatures = function (humidityFeatures) {
1010   return new Promise((resolve, reject) => {
1011     const students = dataCollection.students.filter(
1012       (student) => student.humidityFeatures === humidityFeatures
1013     );
1014     resolve(students);
1015   });
1016 };
1017 module.exports.getStudentsByAirPressureFeatures = function (airPressureFeatures) {
1018   return new Promise((resolve, reject) => {
1019     const students = dataCollection.students.filter(
1020       (student) => student.airPressureFeatures === airPressureFeatures
1021     );
1022     resolve(students);
1023   });
1024 };
1025 module.exports.getStudentsByLightFeatures = function (lightFeatures) {
1026   return new Promise((resolve, reject) => {
1027     const students = dataCollection.students.filter(
1028       (student) => student.lightFeatures === lightFeatures
1029     );
1030     resolve(students);
1031   });
1032 };
1033 module.exports.getStudentsBySoundFeatures = function (soundFeatures) {
1034   return new Promise((resolve, reject) => {
1035     const students = dataCollection.students.filter(
1036       (student) => student.soundFeatures === soundFeatures
1037     );
1038     resolve(students);
1039   });
1040 };
1041 module.exports.getStudentsBySmellFeatures = function (smellFeatures) {
1042   return new Promise((resolve, reject) => {
1043     const students = dataCollection.students.filter(
1044       (student) => student.smellFeatures === smellFeatures
1045     );
1046     resolve(students);
1047   });
1048 };
1049 module.exports.getStudentsByTasteFeatures = function (tasteFeatures) {
1050   return new Promise((resolve, reject) => {
1051     const students = dataCollection.students.filter(
1052       (student) => student.tasteFeatures === tasteFeatures
1053     );
1054     resolve(students);
1055   });
1056 };
1057 module.exports.getStudentsByTouchFeatures = function (touchFeatures) {
1058   return new Promise((resolve, reject) => {
1059     const students = dataCollection.students.filter(
1060       (student) => student.touchFeatures === touchFeatures
1061     );
1062     resolve(students);
1063   });
1064 };
1065 module.exports.getStudentsByTemperatureFeatures = function (temperatureFeatures) {
1066   return new Promise((resolve, reject) => {
1067     const students = dataCollection.students.filter(
1068       (student) => student.temperatureFeatures === temperatureFeatures
1069     );
1070     resolve(students);
1071   });
1072 };
1073 module.exports.getStudentsByHumidityFeatures = function (humidityFeatures) {
1074   return new Promise((resolve, reject) => {
1075     const students = dataCollection.students.filter(
1076       (student) => student.humidityFeatures === humidityFeatures
1077     );
1078     resolve(students);
1079   });
1080 };
1081 module.exports.getStudentsByAirPressureFeatures = function (airPressureFeatures) {
1082   return new Promise((resolve, reject) => {
1083     const students = dataCollection.students.filter(
1084       (student) => student.airPressureFeatures === airPressureFeatures
1085     );
1086     resolve(students);
1087   });
1088 };
1089 module.exports.getStudentsByLightFeatures = function (lightFeatures) {
1090   return new Promise((resolve, reject) => {
1091     const students = dataCollection.students.filter(
1092       (student) => student.lightFeatures === lightFeatures
1093     );
1094     resolve(students);
1095   });
1096 };
1097 module.exports.getStudentsBySoundFeatures = function (soundFeatures) {
1098   return new Promise((resolve, reject) => {
1099     const students = dataCollection.students.filter(
1100       (student) => student.soundFeatures === soundFeatures
1101     );
1102     resolve(students);
1103   });
1104 };
1105 module.exports.getStudentsBySmellFeatures = function (smellFeatures) {
1106   return new Promise((resolve, reject) => {
1107     const students = dataCollection.students.filter(
1108       (student) => student.smellFeatures === smellFeatures
1109     );
1110     resolve(students);
1111   });
1112 };
1113 module.exports.getStudentsByTasteFeatures = function (tasteFeatures) {
1114   return new Promise((resolve, reject) => {
1115     const students = dataCollection.students.filter(
1116       (student) => student.tasteFeatures === tasteFeatures
1117     );
1
```

WEB700\_Assignment5\_Asintha Dilshan Jayasekara (170388235)

The screenshot shows the VS Code editor interface. The Explorer sidebar on the left displays the project structure, including folders like `node_modules` and files like `package-lock.json`. The main editor area shows the content of `collegeData.js`, which is a JavaScript module. The code defines a module with functions to get course by ID, update a student, and write to a JSON file. The Outline and Timeline panels are visible at the bottom left.

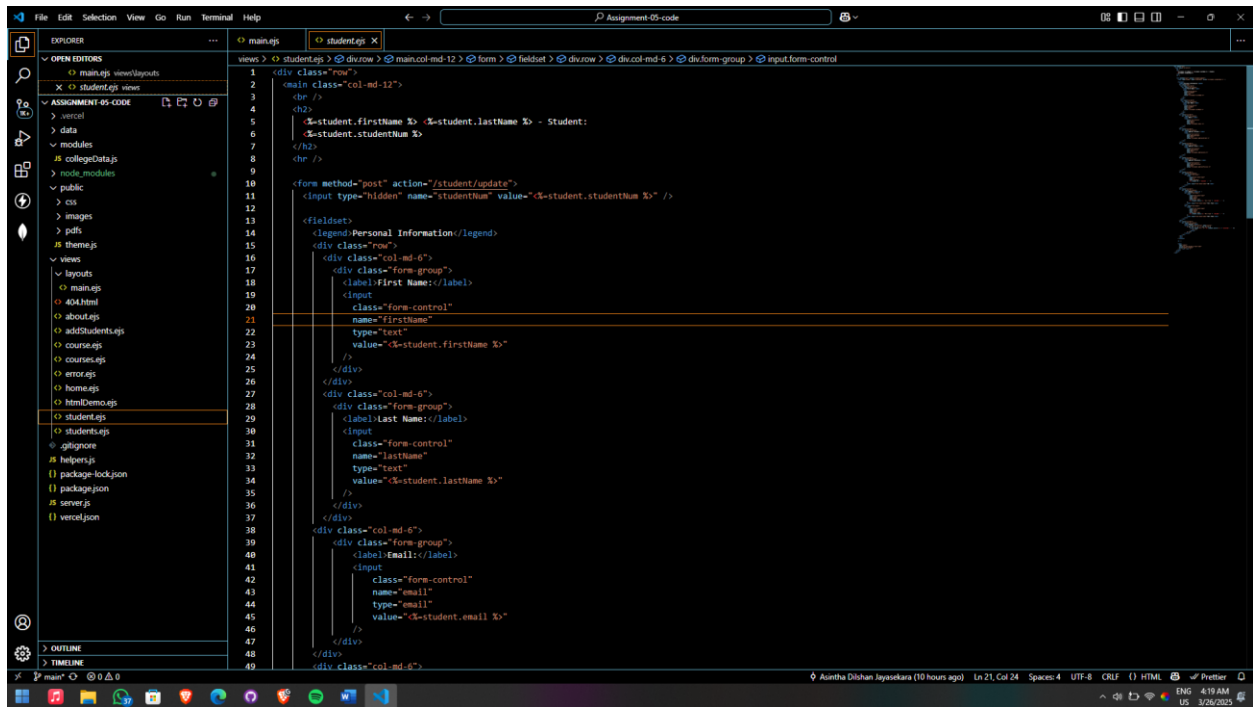
```

181 module.exports.getCourseById = function (id) {
182   // ...
183 }
184
185 module.exports.updateStudent = function (studentData) {
186   return new Promise((resolve, reject) => {
187     if (!dataCollection) {
188       reject("Data not initialized");
189     }
190
191     let found = false;
192
193     for (let i = 0; i < dataCollection.students.length; i++) {
194       if (dataCollection.students[i].studentNum == studentData.studentNum) {
195         studentData.isA = studentData.isA ? true : false;
196         dataCollection.students[i] = studentData;
197         found = true;
198         break;
199       }
200     }
201
202     if (!found) {
203       reject("Student not found");
204     }
205
206     const filePath = path.join(__dirname, "../data/students.json");
207
208     fs.writeFile(
209       filePath,
210       JSON.stringify(dataCollection.students, null, 4),
211       (err) => {
212         if (err) {
213           reject("Unable to write to students.json: " + err);
214         } else {
215           resolve();
216         }
217       }
218     );
219   });
220 }
  
```

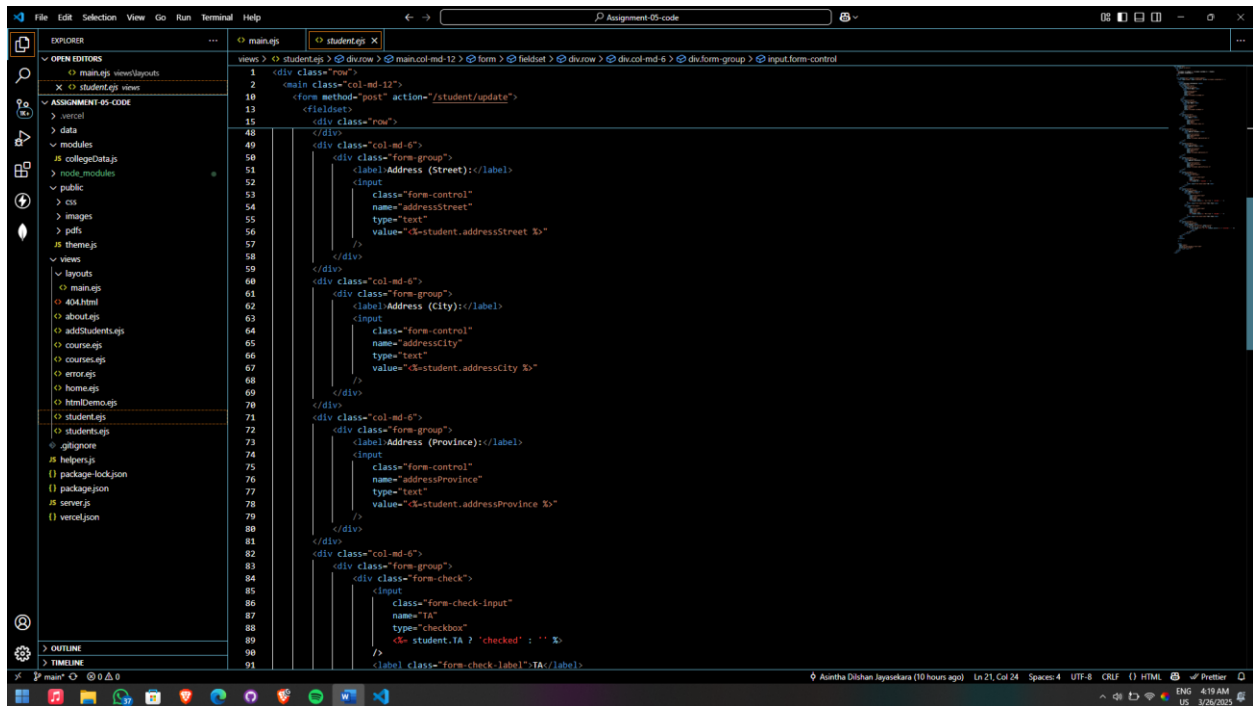
**Students.ejs**

**Student.ejs**

## WEB700\_Assignment5\_Asintha Dilshan Jayasekara (170388235)



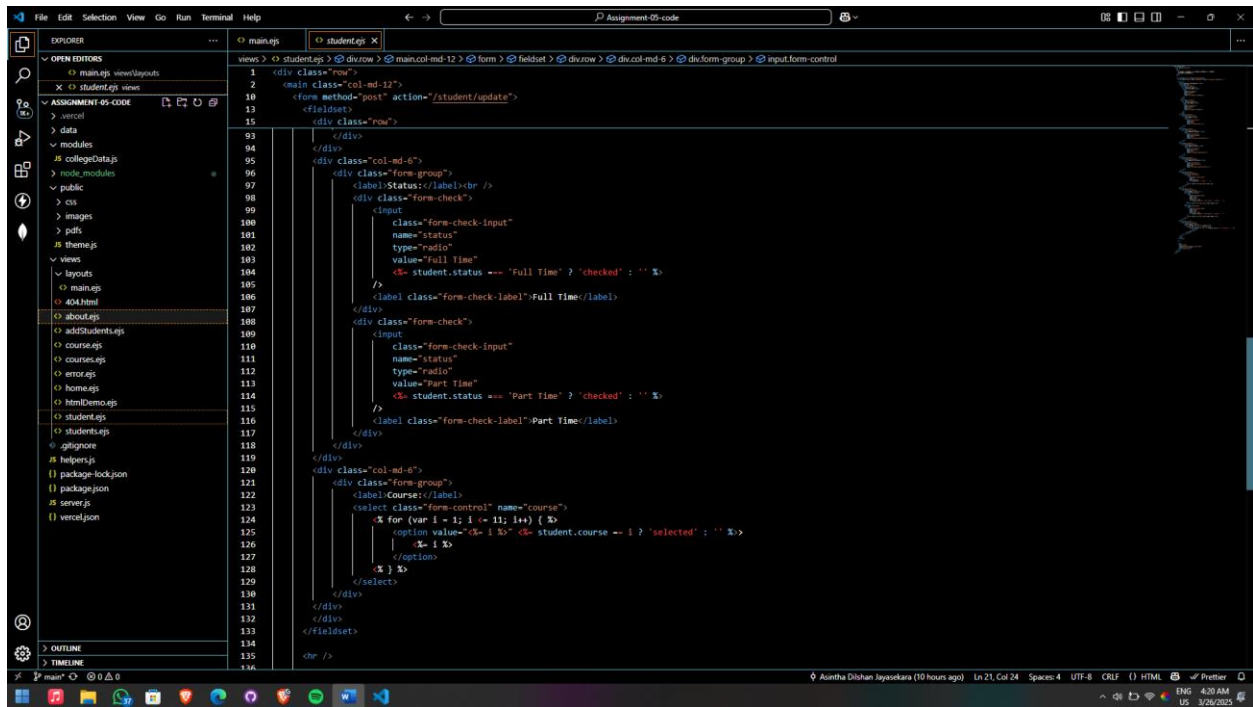
```
1 <div class="row">
2   <main class="col-md-12">
3     <hr />
4     <h2>
5       <student.firstName %s - %student.lastName %s - Student:
6       <student.studentNum %s
7     </h2>
8     <hr />
9
10    <form method="post" action="/student/update">
11      <input type="hidden" name="studentNum" value="%student.studentNum %s" />
12
13      <fieldset>
14        <legend>Personal Information</legend>
15        <div class="row">
16          <div class="col-md-6">
17            <div class="form-group">
18              <label>First Name:</label>
19              <input
20                class="form-control"
21                name="firstName"
22                type="text"
23                value="%student.firstName %s"
24              />
25            </div>
26          </div>
27          <div class="col-md-6">
28            <div class="form-group">
29              <label>Last Name:</label>
30              <input
31                class="form-control"
32                name="lastName"
33                type="text"
34                value="%student.lastName %s"
35              />
36            </div>
37          </div>
38          <div class="col-md-6">
39            <div class="form-group">
40              <label>Email:</label>
41              <input
42                class="form-control"
43                name="email"
44                type="email"
45                value="%student.email %s"
46              />
47            </div>
48          </div>
49        </div>
50      </div>
51    </main>
52  </div>
53 </div>
```



```
1 <div class="row">
2   <main class="col-md-12">
3     <hr />
4     <h2>
5       <student.firstName %s - %student.lastName %s - Student:
6       <student.studentNum %s
7     </h2>
8     <hr />
9
10    <form method="post" action="/student/update">
11      <input type="hidden" name="studentNum" value="%student.studentNum %s" />
12
13      <fieldset>
14        <div class="row">
15          <div class="col-md-6">
16            <div class="form-group">
17              <label>Address (Street):</label>
18              <input
19                class="form-control"
20                name="addressStreet"
21                type="text"
22                value="%student.addressStreet %s"
23              />
24            </div>
25          </div>
26          <div class="col-md-6">
27            <div class="form-group">
28              <label>Address (City):</label>
29              <input
30                class="form-control"
31                name="addressCity"
32                type="text"
33                value="%student.addressCity %s"
34              />
35            </div>
36          </div>
37          <div class="col-md-6">
38            <div class="form-group">
39              <label>Address (Province):</label>
40              <input
41                class="form-control"
42                name="addressProvince"
43                type="text"
44                value="%student.addressProvince %s"
45              />
46            </div>
47          </div>
48          <div class="col-md-6">
49            <div class="form-group">
50              <input
51                class="form-check-input"
52                name="isA"
53                type="checkbox"
54                checked=""
55              />
56              <student.isA ? checked : '' %s
57            </div>
58          </div>
59        </div>
60      </fieldset>
61    </form>
62  </main>
63 </div>
64 </div>
```

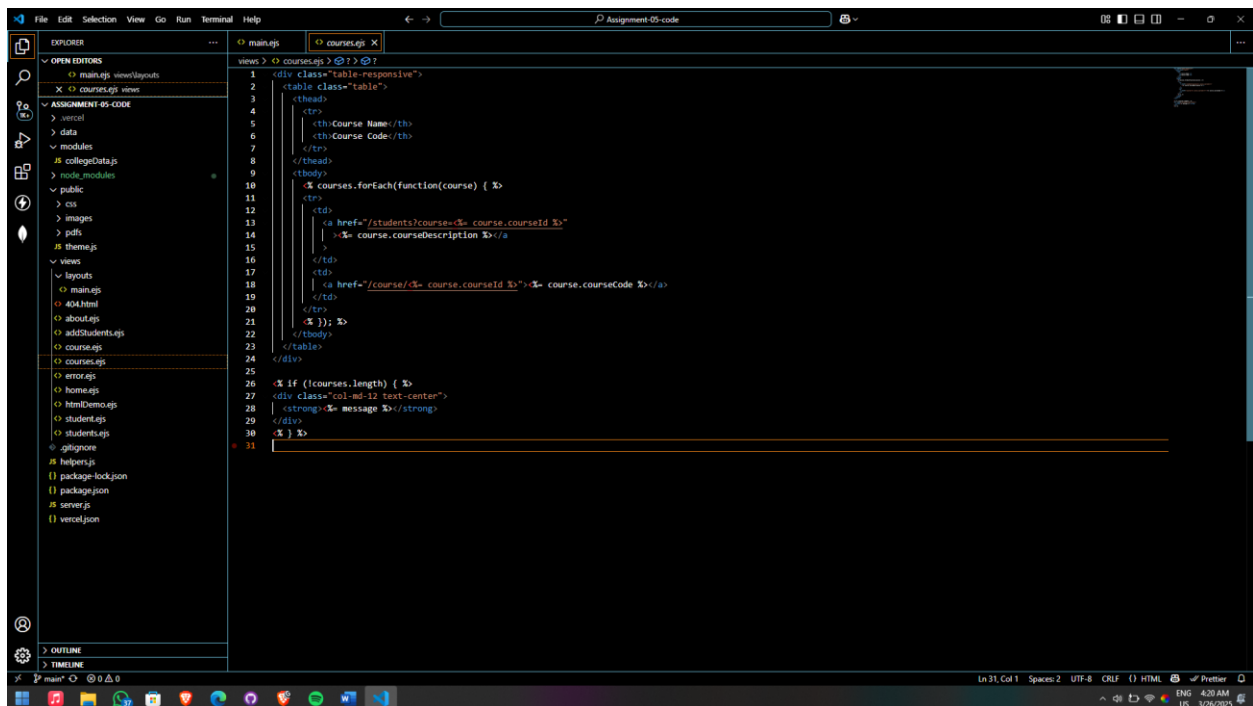


## WEB700\_Assignment5\_Asintha Dilshan Jayasekara (170388235)



```
1 <div class="row">
2   <div class="col-md-12">
3     <form method="post" action="/student/update">
4       <div class="form-group">
5         <input type="text" value="Student Name" />
6       </div>
7       <div class="form-group">
8         <input type="text" value="Student Email" />
9       </div>
10      <div class="form-group">
11        <input type="text" value="Student Phone" />
12      </div>
13      <div class="form-group">
14        <input type="text" value="Student Address" />
15      </div>
16      <div class="form-group">
17        <input type="text" value="Student City" />
18      </div>
19      <div class="form-group">
20        <input type="text" value="Student State" />
21      </div>
22      <div class="form-group">
23        <input type="text" value="Student Zip" />
24      </div>
25      <div class="form-group">
26        <input type="text" value="Student Password" />
27      </div>
28      <div class="form-group">
29        <input type="text" value="Student Confirm Password" />
30      </div>
31      <div class="form-group">
32        <input type="button" value="Update Student" />
33      </div>
34    </div>
35  </div>
36</div>
```

## courses.ejs

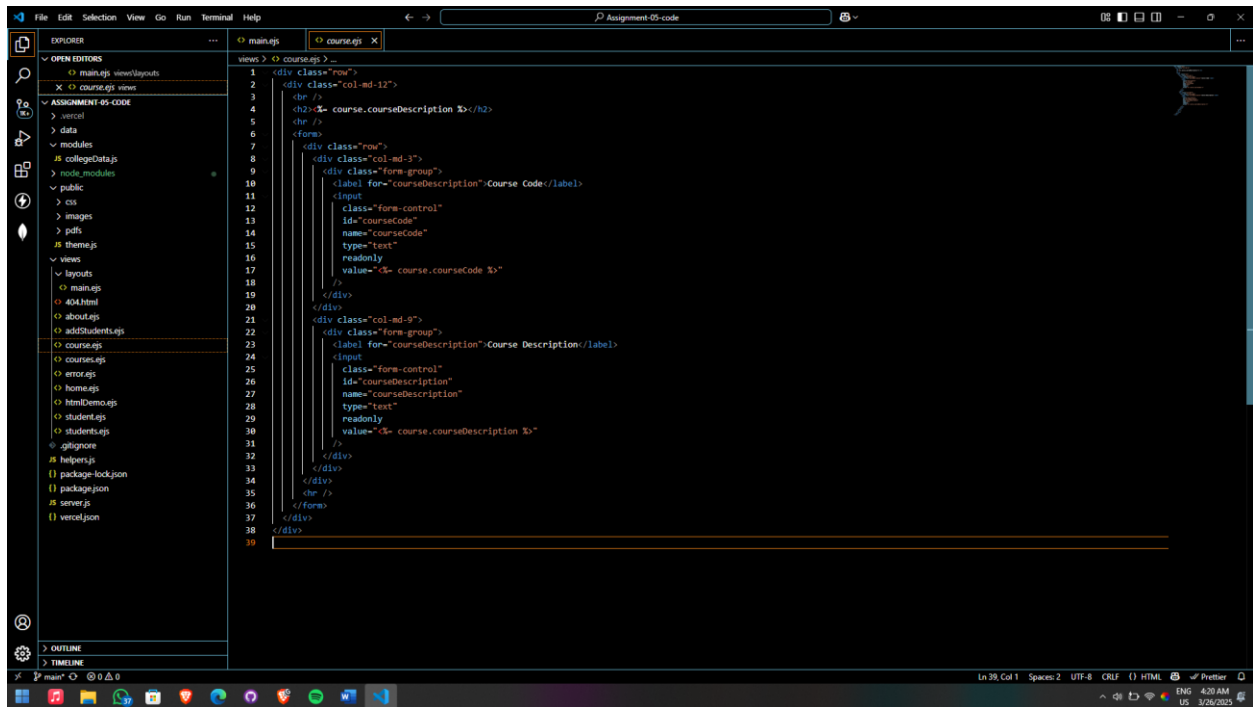


```
1 <div class="table-responsive">
2   <table class="table">
3     <thead>
4       <tr>
5         <th>Course Name</th>
6         <th>Course Code</th>
7       </tr>
8     </thead>
9     <tbody>
10      <tr>
11        <td>
12          <a href="/students/course-1" >
13            <%= course.courseCode %>
14          </a>
15        </td>
16        <td>
17          <a href="/course-1" >
18            <%= course.courseCode %>
19          </a>
20        </td>
21      </tr>
22    </tbody>
23  </table>
24</div>
25
26 <%= if (courses.length) { %>
27   <div class="col-md-12 text-center">
28     <strong><%= message %></strong>
29   </div>
30 <%= } %>
```

## course.ejs



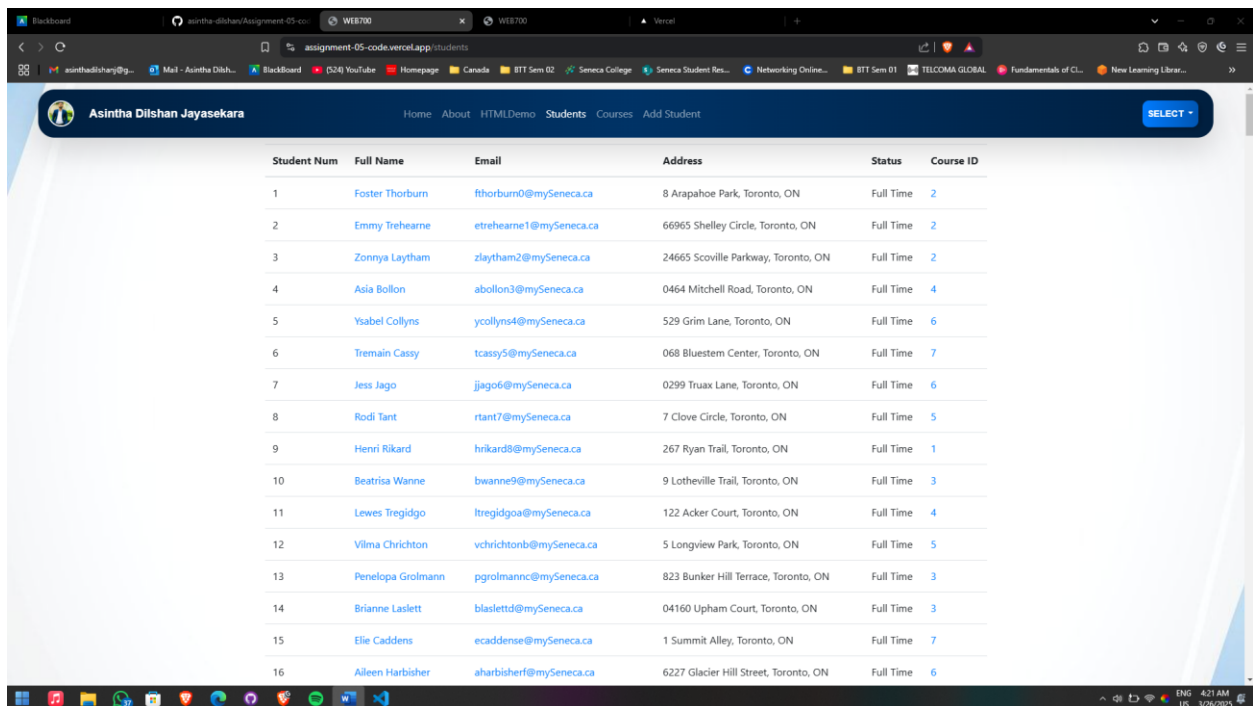
## WEB700\_Assignment5\_Asintha Dilshan Jayasekara (170388235)



The screenshot shows the Visual Studio Code editor with the file explorer on the left displaying the project structure. The main editor window shows the `course.js` file, which contains the following HTML code:

```
1 <div class="row">
2   <div class="col-md-12">
3     <hr />
4     <h2><%= course.courseDescription %></h2>
5     <hr />
6     <form>
7       <div class="row">
8         <div class="col-md-3">
9           <div class="form-group">
10            <label for="courseCode">Course Code</label>
11            <input
12              class="form-control"
13              id="courseCode"
14              name="courseCode"
15              type="text"
16              readonly
17              value="<%= course.courseCode %>"
18            />
19          </div>
20        </div>
21        <div class="col-md-9">
22          <div class="form-group">
23            <label for="courseDescription">Course Description</label>
24            <input
25              class="form-control"
26              id="courseDescription"
27              name="courseDescription"
28              type="text"
29              readonly
30              value="<%= course.courseDescription %>"
31            />
32          </div>
33        </div>
34      </div>
35    </div>
36  </div>
37 </div>
38
39
```

## Final Output



The screenshot shows the web application running in a browser. The page displays a table of student information. The table has the following columns: Student Num, Full Name, Email, Address, Status, and Course ID. The data is as follows:

Student Num	Full Name	Email	Address	Status	Course ID
1	Foster Thorburn	fthorburn0@mySeneca.ca	8 Arapahoe Park, Toronto, ON	Full Time	2
2	Emmy Trehearne	etrehearne1@mySeneca.ca	66965 Shelley Circle, Toronto, ON	Full Time	2
3	Zonnya Laytham	zlaytham2@mySeneca.ca	24665 Scoville Parkway, Toronto, ON	Full Time	2
4	Asia Bollon	abollon3@mySeneca.ca	0464 Mitchell Road, Toronto, ON	Full Time	4
5	Ysabel Collins	ycollins4@mySeneca.ca	529 Grim Lane, Toronto, ON	Full Time	6
6	Tremain Cassy	tcassy5@mySeneca.ca	068 Bluestem Center, Toronto, ON	Full Time	7
7	Jess Jago	jjago6@mySeneca.ca	0299 Truax Lane, Toronto, ON	Full Time	6
8	Rodi Tant	rtant7@mySeneca.ca	7 Clove Circle, Toronto, ON	Full Time	5
9	Henri Rikard	hrikard8@mySeneca.ca	267 Ryan Trail, Toronto, ON	Full Time	1
10	Beatrisa Wanne	bwanne9@mySeneca.ca	9 Lothville Trail, Toronto, ON	Full Time	3
11	Lewes Tregidgo	ltregidgoa@mySeneca.ca	122 Acker Court, Toronto, ON	Full Time	4
12	Vilma Chrichton	vchrichtonb@mySeneca.ca	5 Longview Park, Toronto, ON	Full Time	5
13	Penelopa Grolmann	pgrolmannc@mySeneca.ca	823 Bunker Hill Terrace, Toronto, ON	Full Time	3
14	Brianne Laslett	blaslett@mySeneca.ca	04160 Upham Court, Toronto, ON	Full Time	3
15	Elie Caddens	ecaddensed@mySeneca.ca	1 Summit Alley, Toronto, ON	Full Time	7
16	Aileen Harbisher	aharbisherf@mySeneca.ca	6227 Glacier Hill Street, Toronto, ON	Full Time	6

## WEB700\_Assignment5\_Asintha Dilshan Jayasekara (170388235)

53	Goldina Farres	gfarresm@mySeneca.ca	472 Esch Court, Toronto, ON	Full Time	5
54	Karlen Keam	kkeamn@mySeneca.ca	9300 Velth Hill, Toronto, ON	Full Time	7
55	Maxie Loneragan	mlonerano@mySeneca.ca	08 Karstens Trail, Toronto, ON	Full Time	2
56	Virgil Youle	vyoulep@mySeneca.ca	02 Scofield Way, Toronto, ON	Full Time	1
57	Alexi Fish	afishq@mySeneca.ca	079 Northwestern Pass, Toronto, ON	Full Time	4
58	Keefe De Mattia	kdemattiar@mySeneca.ca	54 Moose Plaza, Toronto, ON	Full Time	3
59	Mano Hooke	mhooks@mySeneca.ca	750 Sutteridge Place, Toronto, ON	Full Time	3
60	Tiffanie Ferrarotti	tferrarotti@mySeneca.ca	8360 Leirdahl Crossing, Toronto, ON	Part Time	2
61	Valle Joberne	vjoberneu@mySeneca.ca	885 Sommers Hill, Toronto, ON	Full Time	4
62	Bevon Elce	belcev@mySeneca.ca	68665 Fieldstone Terrace, Toronto, ON	Part Time	6
63	Mill Cutill	mcutillw@mySeneca.ca	4333 Corben Place, Toronto, ON	Full Time	5
64	Aurelea Cleyburn	acleburnx@mySeneca.ca	2 Schurz Circle, Toronto, ON	Part Time	3
65	Shaina Lingner	slingnery@mySeneca.ca	8739 International Terrace, Toronto, ON	Full Time	6
66	Nancee Delicate	ndelicatez@mySeneca.ca	581 Fremont Lane, Toronto, ON	Full Time	7
67	Olenka Mountfort	omountfort10@mySeneca.ca	45 Crest Line Hill, Toronto, ON	Full Time	5
68	Bianca Biesty	bbiesty11@mySeneca.ca	9 Mosinee Way, Toronto, ON	Part Time	7
69	Jedediah Mariner	jmariner12@mySeneca.ca	941 Union Point, Toronto, ON	Full Time	6
70	Obie Rubinovici	orubinovici13@mySeneca.ca	4497 Lunder Crossing, Toronto, ON	Part Time	1
71	Brit Bresson	bbresson14@mySeneca.ca	74 Thierer Street, Toronto, ON	Full Time	2

244	Aura Esmead	aesmead5x@mySeneca.ca	24232 Wayridge Trail, Toronto, ON	Full Time	6
245	Ernie Birth	ebirth5y@mySeneca.ca	15060 Meadow Ridge Drive, Toronto, ON	Part Time	4
246	Stanwood Phillips	sphillips5z@mySeneca.ca	68 Johnson Parkway, Toronto, ON	Full Time	2
247	Raquel Sign	rsign60@mySeneca.ca	74 High Crossing Avenue, Toronto, ON	Full Time	7
248	Christian Tarbet	ctarbet61@mySeneca.ca	7 Debra Alley, Toronto, ON	Part Time	6
249	Cassy Colquyte	ccolquyte62@mySeneca.ca	7745 Sachs Road, Toronto, ON	Full Time	7
250	Gale Davidowsky	gdavidowsky63@mySeneca.ca	1 Alpine Place, Toronto, ON	Full Time	3
251	Sascha Barhems	sbarhems64@mySeneca.ca	9708 Dixon Alley, Toronto, ON	Part Time	1
252	Cristionna Beeching	cbeeching65@mySeneca.ca	7865 Sullivan Center, Toronto, ON	Full Time	2
253	Binny Casely	bcasely66@mySeneca.ca	16 Clemoris Plaza, Toronto, ON	Full Time	3
254	Gnni Giovannacc@i	ggiovannacc67@mySeneca.ca	729 Parkside Road, Toronto, ON	Full Time	3
255	Ignaz Saintpierre	isaintpierre68@mySeneca.ca	3190 Fairfield Road, Toronto, ON	Full Time	2
256	Penny Souza	psouza69@mySeneca.ca	8557 Lindbergh Hill, Toronto, ON	Full Time	4
257	Lothaire Aveling	laveling6a@mySeneca.ca	60 Kennedy Place, Toronto, ON	Part Time	6
258	Abramo Cecil	acecil6b@mySeneca.ca	920 Anthes Parkway, Toronto, ON	Full Time	2
259	Tessy Verry	tverry6c@mySeneca.ca	8142 School Plaza, Toronto, ON	Part Time	6
260	Jojo Tschierasche	jtschierasche6d@mySeneca.ca	0560 Di Loreto Crossing, Toronto, ON	Part Time	5
261	Asintha Jayasekara	adjwisurumana-arachc@myseneca.ca	1485 Birchmount Road, Scarborough, ON	Full Time	1

## WEB700\_Assignment5\_Asintha Dilshan Jayasekara (170388235)

The screenshot shows a web application interface for a user named Asintha Dilshan Jayasekara. The top navigation bar includes links for Home, About, HTMLDemo, Students, Courses, and Add Student, along with a 'SELECT' dropdown. The main content area displays a table of courses with two columns: 'Course Name' and 'Course Code'. The table lists ten courses, including 'Relational Database Design and Implementation' (DES720), 'Java Programming' (JAV745), 'Introduction to Cloud Computing' (OPS705), 'Database Administration and Management' (SQL710), 'Web Programming' (WEB700), 'Applied Capstone Project' (CAP805), 'Database Connectivity Using Java' (CJV805), 'Accessing Big Data' (DBD800), 'Datawarehousing' (DBW825), and 'Security in Databases and Web Applications' (SEC835). A final row for 'Work Term Preparation (Work-Integrated Learning option only)' has a course code of WTP100. The browser's address bar shows the URL 'assignment-05-code.vercel.app/courses'.

Course Name	Course Code
Relational Database Design and Implementation	DES720
Java Programming	JAV745
Introduction to Cloud Computing	OPS705
Database Administration and Management	SQL710
Web Programming	WEB700
Applied Capstone Project	CAP805
Database Connectivity Using Java	CJV805
Accessing Big Data	DBD800
Datawarehousing	DBW825
Security in Databases and Web Applications	SEC835
Work Term Preparation (Work-Integrated Learning option only)	WTP100

The screenshot shows the 'Add a New Student' form in the same web application. The form is titled 'Add a New Student' and is divided into 'Personal Information' and 'Enrollment Status' sections. The 'Personal Information' section includes fields for First Name, Last Name, Email, Address (Street), Address (City), and Address (Province). The 'Enrollment Status' section includes a checkbox for 'Teaching Assistant' and radio buttons for 'Full Time' and 'Part Time'. The 'Enrolled in Course' dropdown menu is set to 'Relational Database Design and Implementation'. The 'Add Student' button is located at the bottom right of the form. The browser's address bar shows the URL 'assignment-05-code.vercel.app/addStudents'.

**Add a New Student**

**Personal Information**

First Name:

Last Name:

Email:

Address (Street):

Address (City):

Address (Province):

☐ Teaching Assistant

Enrollment Status:

☐ Full Time

☐ Part Time

Enrolled in Course: