



Gateway Revamp

Will Gillette, Kacey La, Jessica Sokolski, Matt
Quigley, and Ryan Fitzgerald

Proposal

- What is the issue?
- Our Mission
 - Create a more efficient system that resolves all these issues, adds additional features, and is more user-friendly.
- We hope to streamline the current system and ultimately provide students a more productive experience when planning out their degrees.

Course Details

DIGS-200: Intro Dig Studies

DIGS 200: Introduction to Digital Studies This course cultivates foundational knowledge in digital studies, providing students with an understanding of central questions in the field. In this course, students will examine what it means to be digitally literate. The course will address questions of power, privilege, and access that shape the digital world. Beginning with an historical overview, students will explore issues such as digital security and privacy, digital identities, and copyright and fair use. Along with this theoretical grounding, students will be introduced to key principles for project management and development, data analysis, visualization, and curation. While gaining familiarity with and thinking critically about a variety of tools and platforms, students will build projects that help them develop a context for informed, ethical choices about the use of technology. Required for the digital studies minor. Offered every other year. Three hours a week. Four semester hours. (LINQ.)

What year?

Course Description

CS-373. Theory of Computation Principles of formal languages, automata, computability and computational complexity. Emphasis on writing proofs of theorems. Prerequisites: MATH-236W, A grade of C- or higher in CS-174. Offered in the fall of even years. Three hours per week. Four semester hours.

Fall of odd years

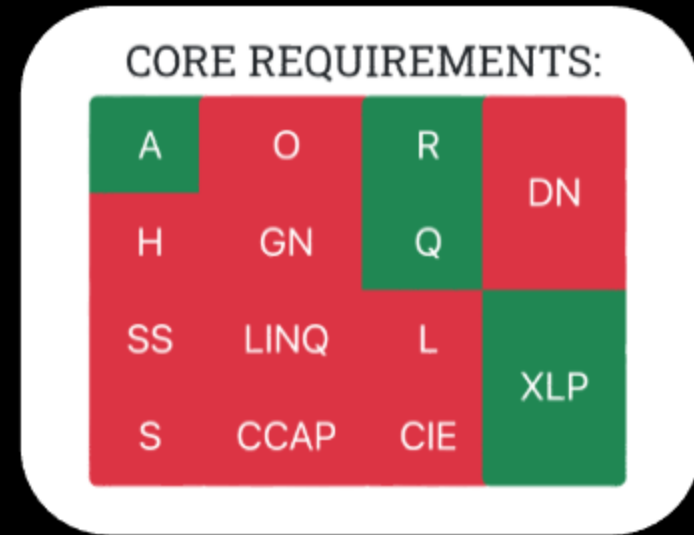
Stakeholders

- UC Students
- Advisors
- Registrar
- Aspirant Stakeholder
- Other Universities/Colleges



Requirements

- Semester/Schedule Planning
 - Select only courses offered that semester
 - Filter for courses
- Course Catalog
 - Easily browse courses
 - See when each course is offered
- Degree Progress
 - Quickly see Fulfilled/Unfulfilled college requirements
 - Mark course with multiple letters



Course Browser

Select Semester:
F2025

☐ FALL ☐ SPRING ☐ EVEN ☐ ODD
☐ A ☐ H ☐ SS ☐ S
☐ O ☐ GN ☐ LINQ ☐ CCAP
☐ R ☐ Q ☐ L ☐ CIE
☐ DN ☐ XLP

Search for a course by inputting its title or id

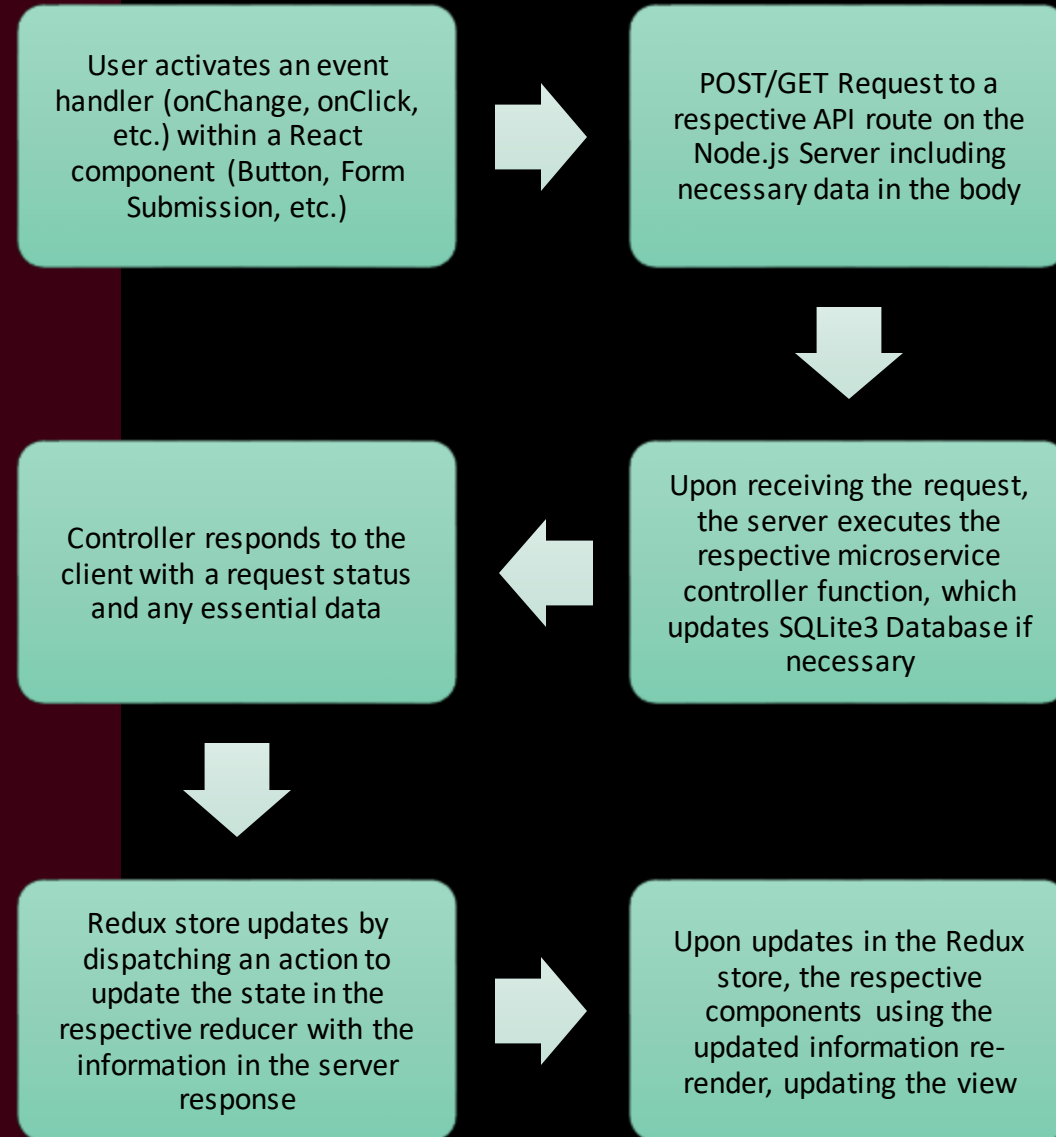
Course List

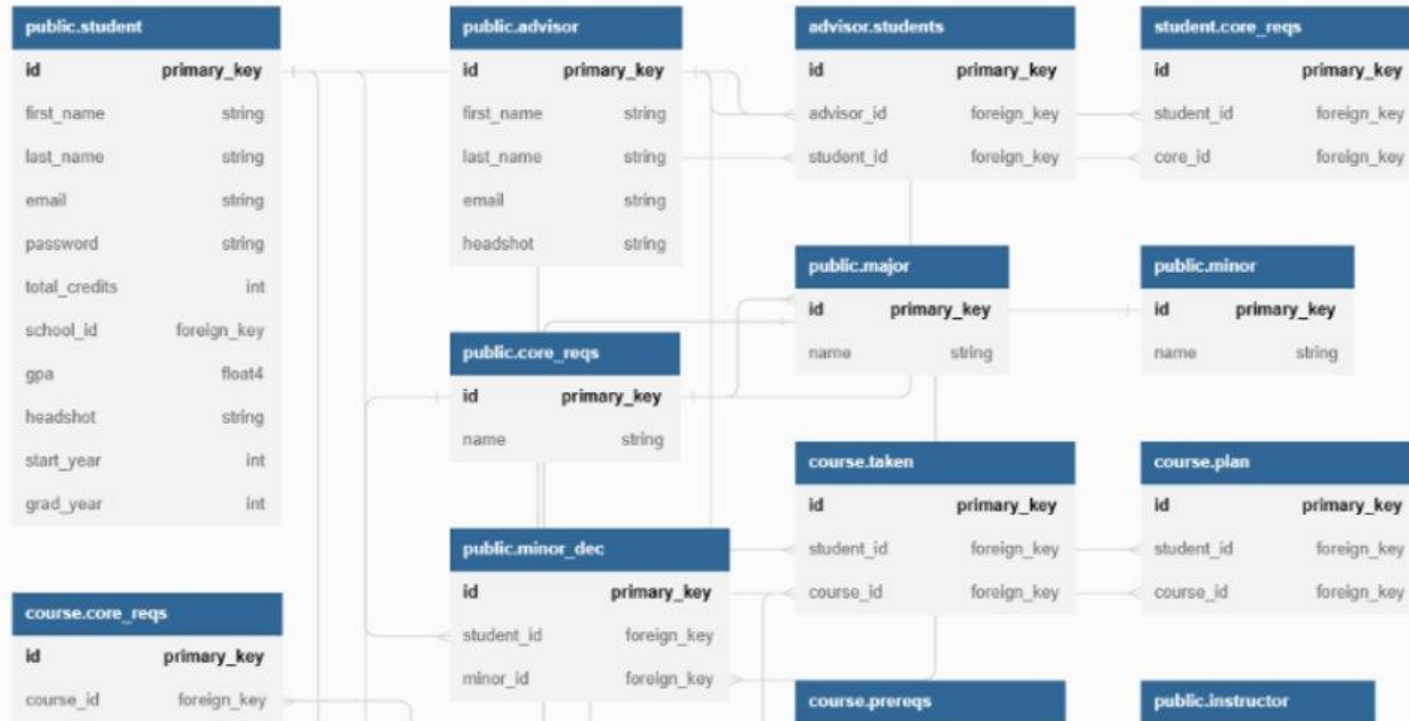
CS-173 <input type="checkbox"/> Select Intro Comp Sci PREREQUISITES: NONE Q, R Credits: 4	CS-176 <input type="checkbox"/> Select OO Programming PREREQUISITES: + CS-173 Credits: 4	CS-336 <input type="checkbox"/> Select Isdep Study PREREQUISITES: NONE XLP Credits: 4
MATH-111 <input type="checkbox"/> Select Calculus I PREREQUISITES: NONE	MATH-112 <input type="checkbox"/> Select Calculus II PREREQUISITES:	MATH-235 <input type="checkbox"/> Select Lower Algebra PREREQUISITES:

ADD COURSE

Design

- SQLite3 Database
- JavaScript
- Node JS
- React
- Redux
- Bootstrap

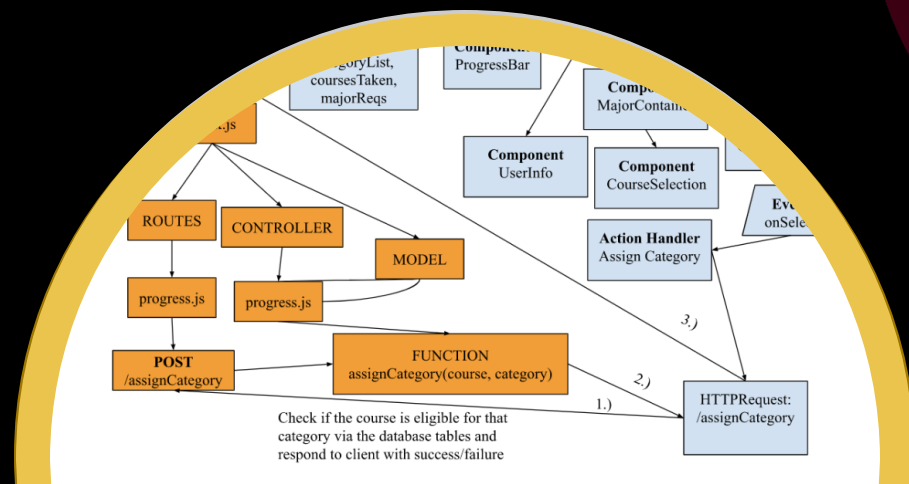
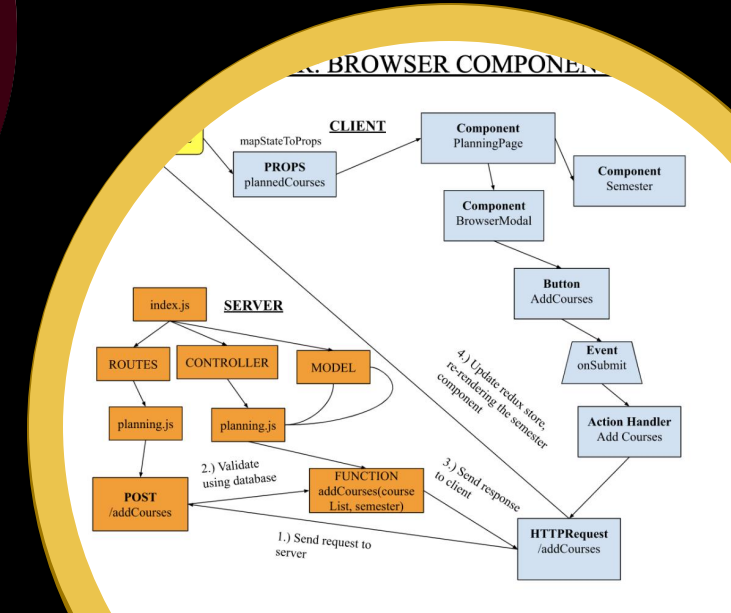
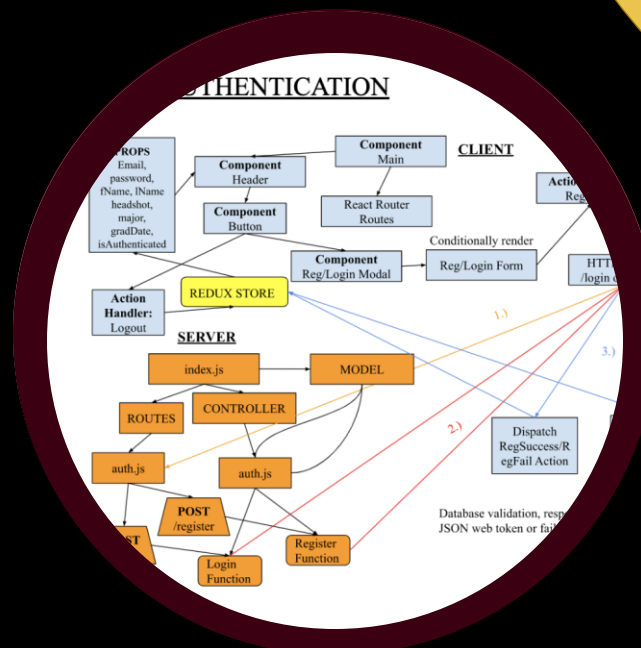
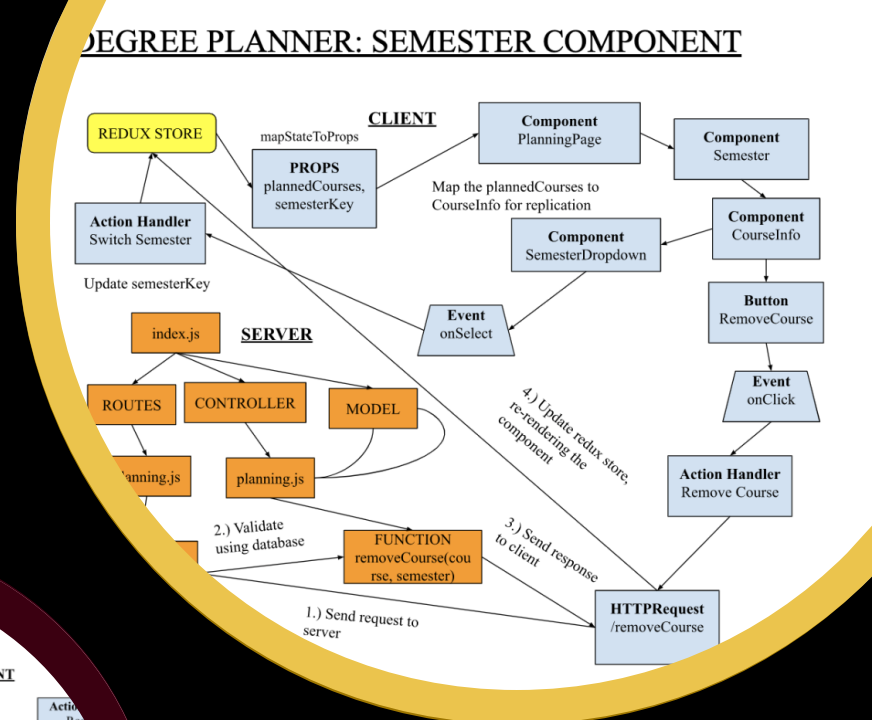
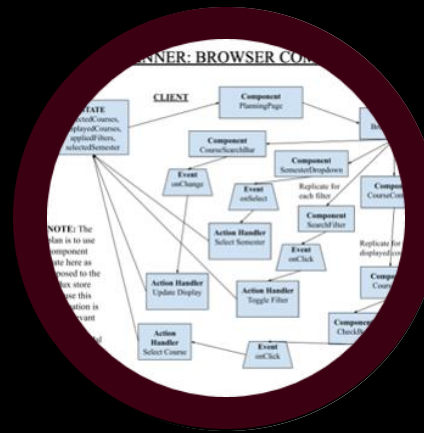




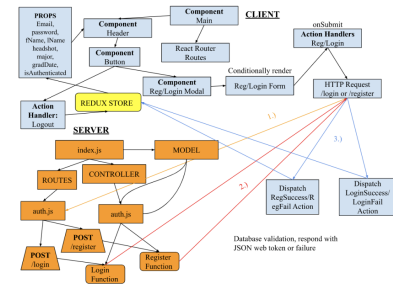
SQLite3 Database

Application

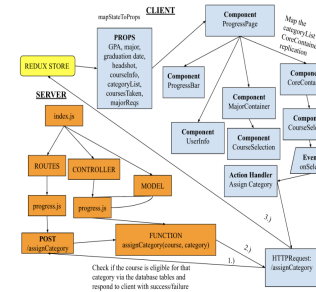
Traces



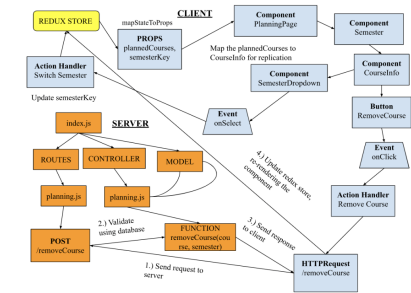
USER AUTHENTICATION



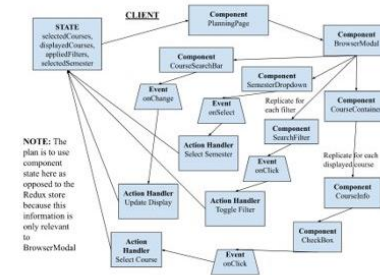
DEGREE PROGRESS



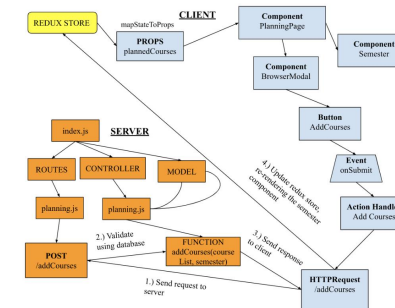
DEGREE PLANNER: SEMESTER COMPONENT



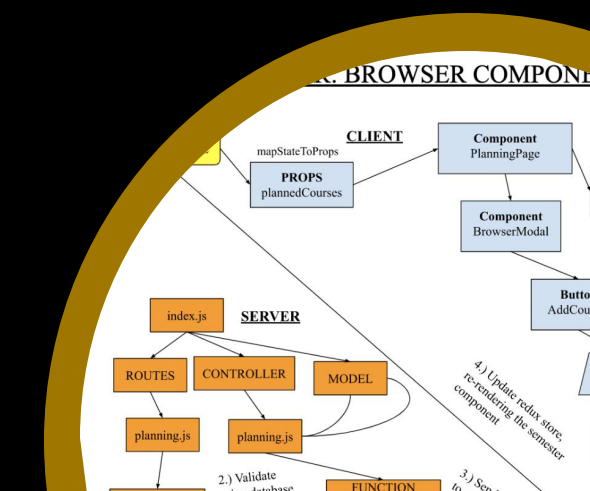
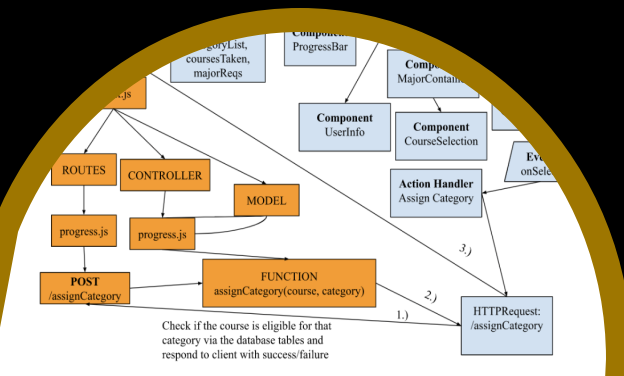
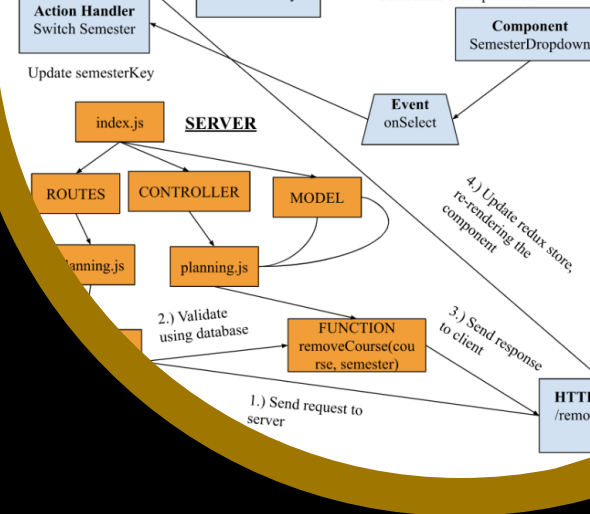
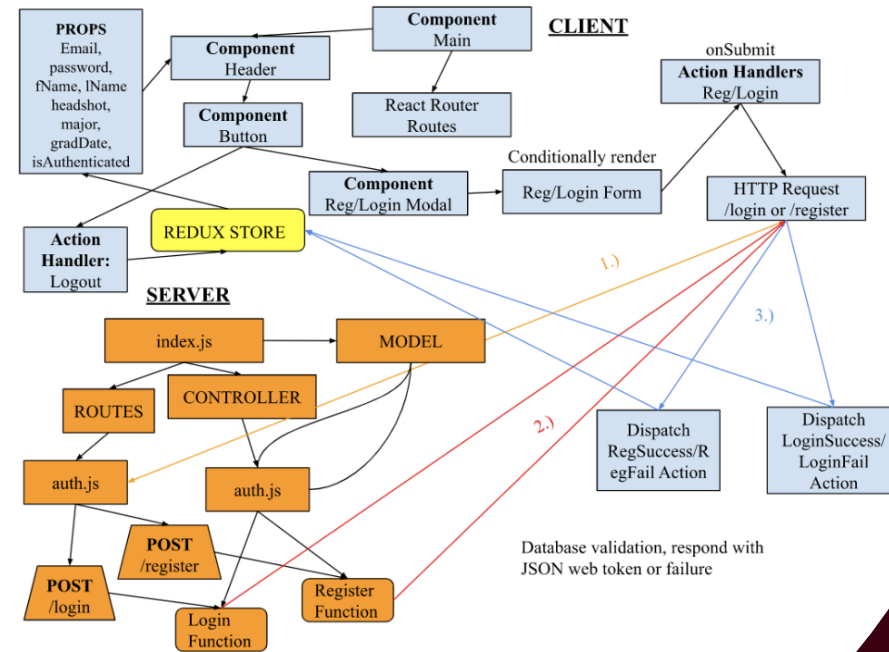
DEGREE PLANNER: BROWSER COMPONENT #1



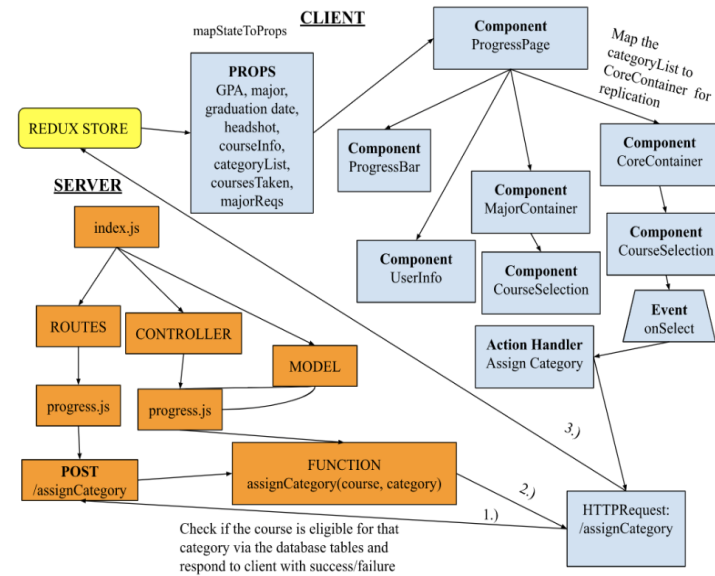
DEGREE PLANNER: BROWSER COMPONENT #2



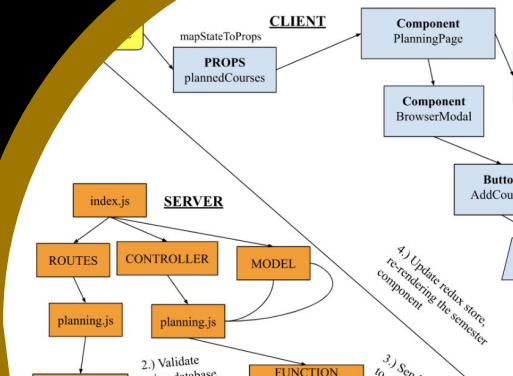
USER AUTHENTICATION



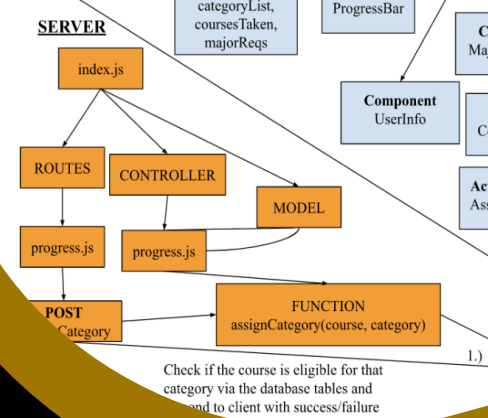
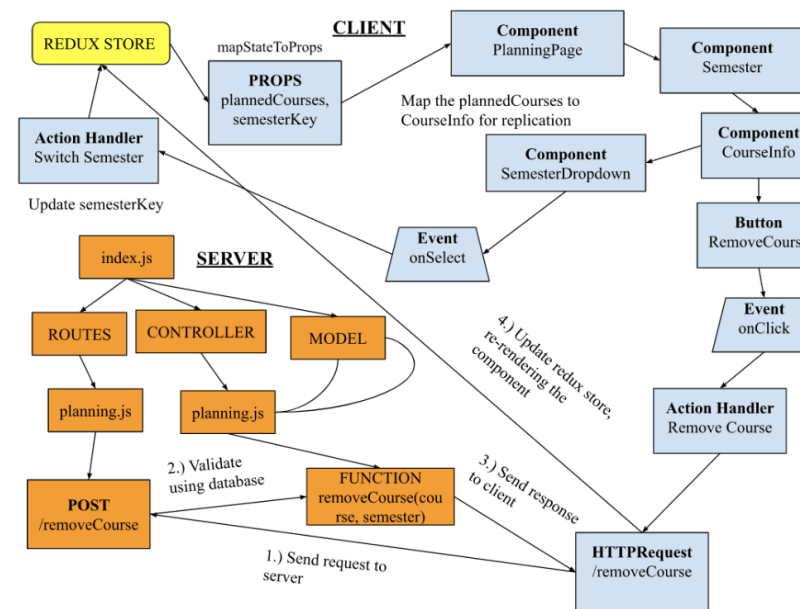
DEGREE PROGRESS



BROWSER COMPONENT



DEGREE PLANNER: SEMESTER COMPONENT



The diagram illustrates the flow of data and control between the CLIENT and SERVER components.

CLIENT Side:

- Component PlanningPage** (light blue box) receives data from the SERVER via the **mapStateToProps** prop.
- Component BrowserModal** (light blue box) is connected to the **Component PlanningPage**.
- Button AddCourse** (light blue box) is connected to the **Component BrowserModal**.

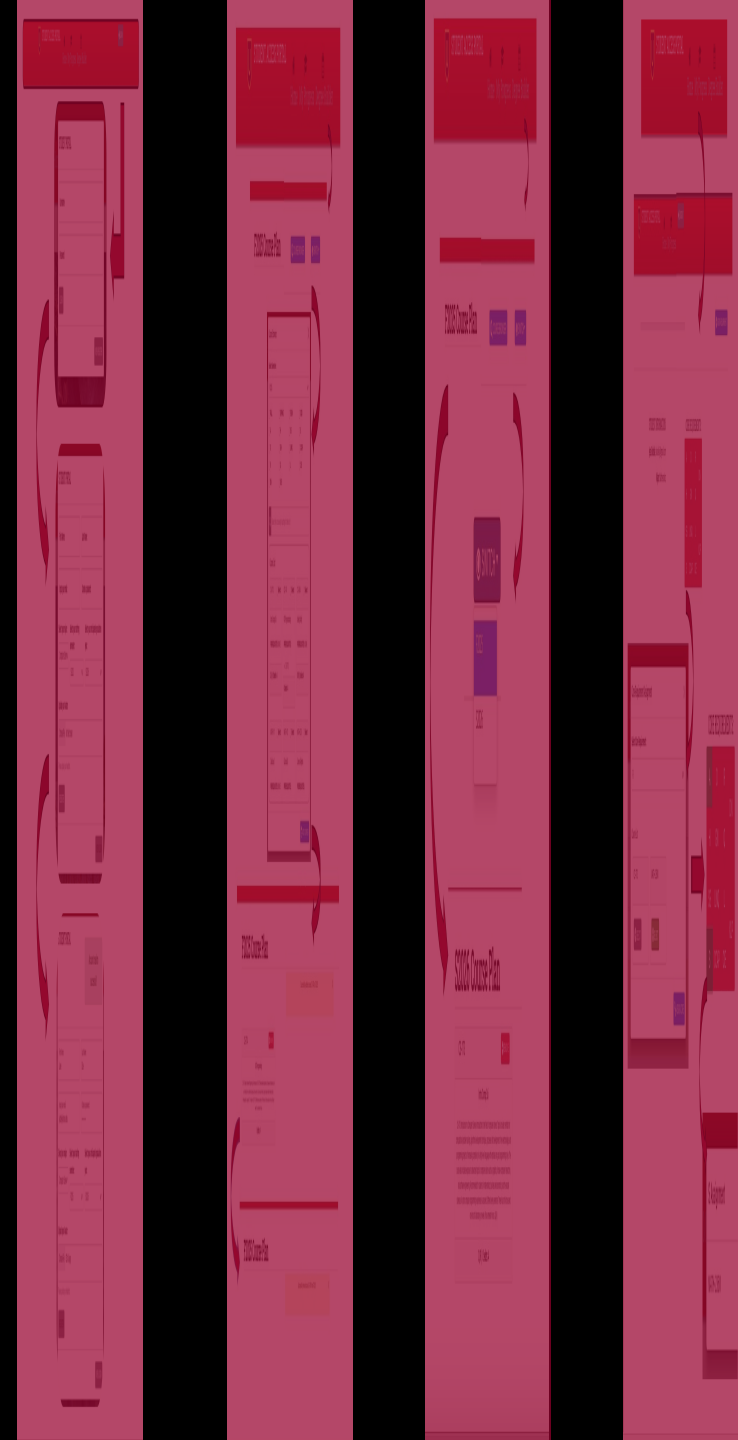
SERVER Side:

- index.js** (orange box) is the main entry point, connecting to **ROUTES**, **CONTROLLER**, and **MODEL**.
- ROUTES** (orange box) connects to **planning.js** (orange box).
- CONTROLLER** (orange box) connects to **planning.js** (orange box).
- MODEL** (orange box) connects to **planning.js** (orange box).
- planning.js** (orange box) is the main logic file, connecting to the **FUNCTION** (orange box).

Annotations:

- CLIENT** (underlined text) is positioned above the client-side components.
- SERVER** (underlined text) is positioned above the server-side components.
- mapStateToProps** (text) is positioned between the **Component PlanningPage** and the **index.js**.
- props** (text) is positioned above the **Component PlanningPage**.
- plannedCourses** (text) is positioned above the **Component PlanningPage**.
- 2.) Validate database** (text) is positioned below the **planning.js** box.
- 3.) Send to database** (text) is positioned below the **planning.js** box.
- 4.) Update redux store, re-rendering the semester component.** (text) is positioned to the right of the **Component PlanningPage**.

Front End Demo



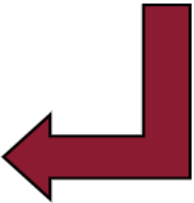
STUDENT PORTAL

Username

Password

LOGIN

VIEW REGISTER



STUDENT PORTAL

First Name:

Last Name:

Input your email:

Create a password:

Select your major: Computer Scien

Select your starting semester: S2023

Select your anticipated graduation year: S2026

Upload your Avatar:

Choose File No file chosen

Please upload your headshot.

REGISTER

VIEW LOGIN

STUDENT PORTAL

Account creation successful!

First Name: Jane

Last Name: Doe

Input your email: jadoe@ursinus.edu

Create a password:

Select your major: Computer Scien

Select your starting semester: F2025

Select your anticipated graduation year: S2026

Upload your Avatar:

Choose File CB fall.jpeg

Please upload your headshot.

REGISTER

VIEW LOGIN

Login & Registration



F2025 Course Plan

[Q COURSE BROWSER](#)[SWITCH](#)

Course Browser

Select Semester:
F2025

☐ FALL
☐ A
☐ O
☐ R
☐ DN

☐ SPRING
☐ H
☐ GN
☐ Q
☐ XLP

☐ EVEN
☐ SS
☐ LING
☐ L

☐ ODD
☐ S
☐ CCAP
☐ CIE

Course List

CS-173 ☐ Select

Intro Comp Sci

PREREQUISITES: NONE

Q, R | Credits: 4

CS-174 ☐ Select

OO Programming

PREREQUISITES:
+ CS-173

Credits: 4

CS-394 ☐ Select

Indep Study

PREREQUISITES: NONE

XLP | Credits: 4

MATH-111 ☐ Select

Calculus I

PREREQUISITES: NONE

MATH-112 ☐ Select

Calculus II

PREREQUISITES:

MATH-235 ☐ Select

Linear Algebra

PREREQUISITES:

[ADD COURSES](#)

F2025 Course Plan

Successfully added courses CS-174 to F2025!

CS-174 [REMOVE](#)

OO Programming

CS-174, Object-Oriented Programming. A continuation of CS-173. More detailed exploration of classes and instances, and an introduction to collection classes such as vectors, lists, maps and sets. Larger programs and/or team projects. Prerequisite: a grade of C- or higher in CS-173. Offered every semester. Three hours of lecture and one hour of lab per week. Four semester hours.

Credits: 4

F2025 Course Plan

Successfully removed course CS-174 from F2025!

Scheduling

F2025 Course Plan

[Q COURSE BROWSER](#)[SWITCH](#)

 SWITCH

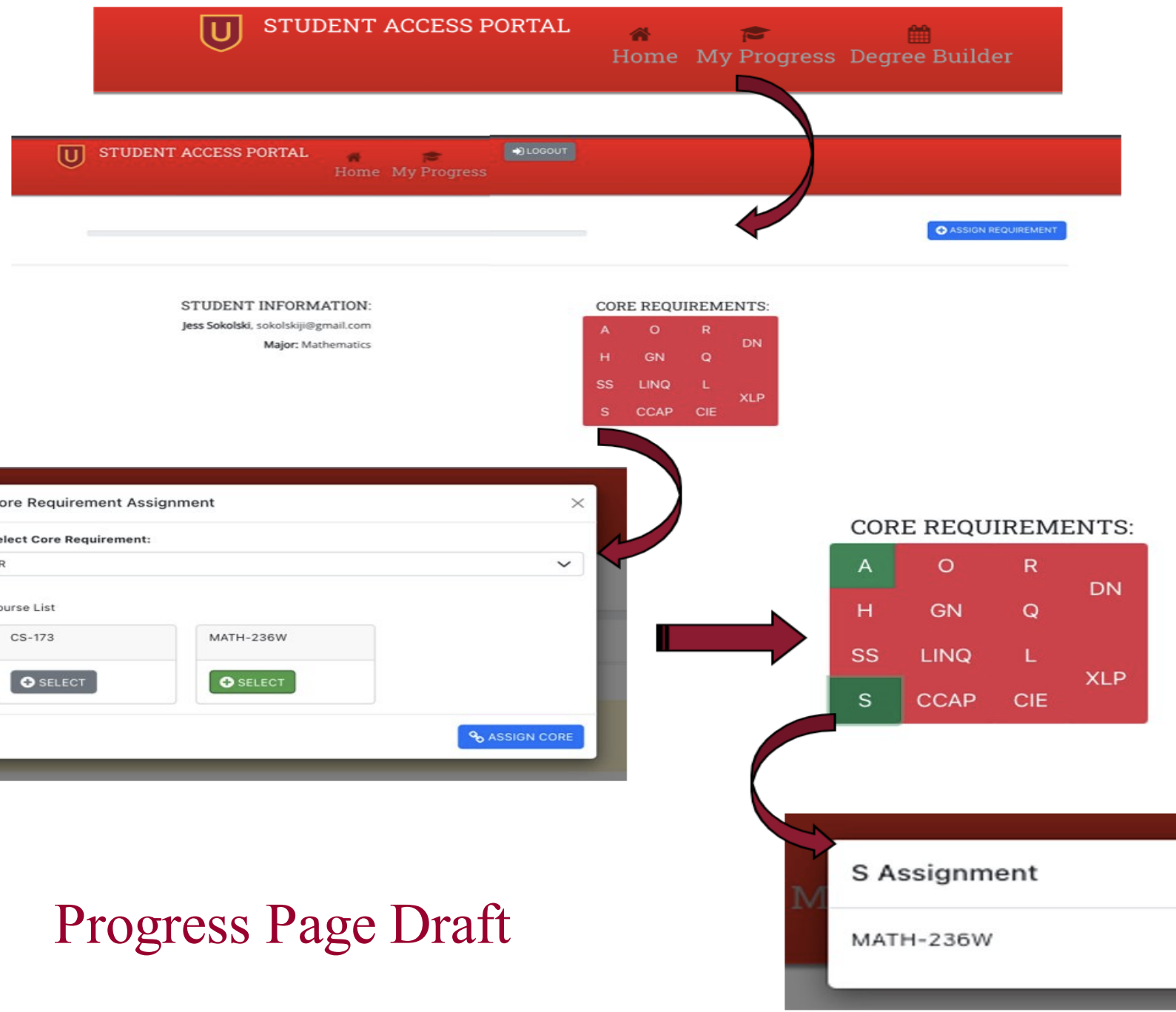
F2025

S2026

S2026 Course Plan

CS-173	REMOVE
Intro Comp Sci	
CS-173. Introduction to Computer Science Introduction to the field of computer science. Topics include: methods for computational problem solving, algorithm development techniques, processes for development of new technologies, and programming projects of increasing complexity in a high-level language with emphasis on good programming style. The course also includes exposure to advanced topics in computer science such as graphics, human-computer interaction, and software engineering. Recommended for students in mathematics, business and economics; and the natural sciences. No prior computer programming experience is assumed. Offered every semester. Three hours of lecture and one hour of laboratory per week. Four semester hours. (Q,R)	
Q, R Credits: 4	

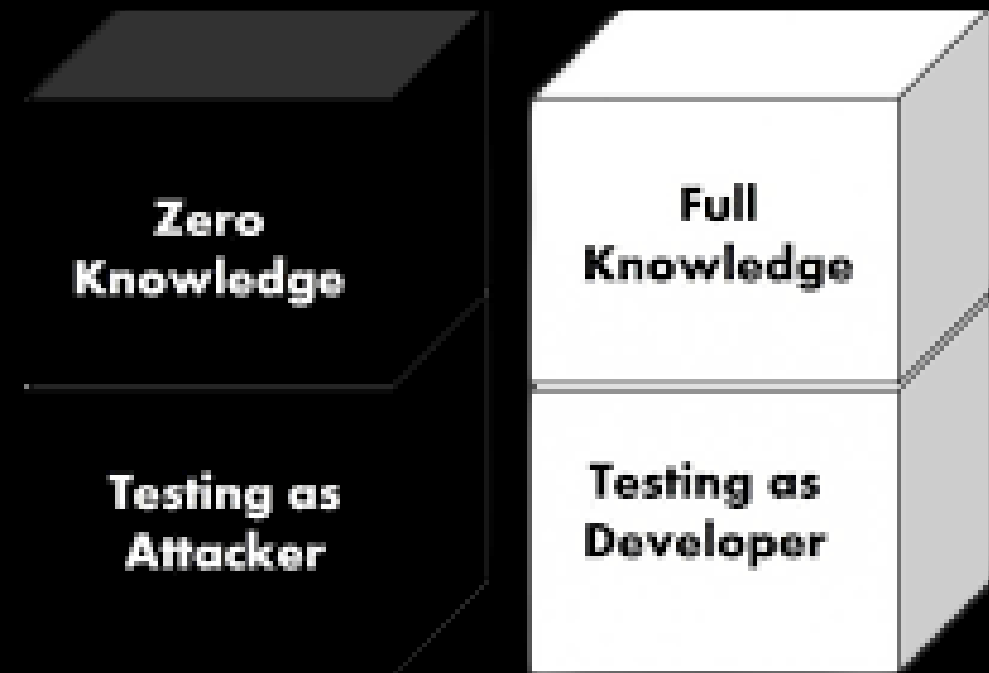
Semester
Planning



Progress Page Draft

Test Plan

- Black Box Testing
- White Box Testing
- User Acceptance Tests



Back End Demo

- Create SQL tables based on the diagram
- Insert JSON course and core requirements data
- Write the server functions for each requirement

```
// Courses
console.log('INIT TABLE COURSES')
db.exec(`
  CREATE TABLE IF NOT EXISTS Courses
  (
    ID TEXT PRIMARY KEY,
    title VARCHAR(50) NOT NULL,
    description VARCHAR(256) NOT NULL,
    creditAmount INTEGER NOT NULL,
    location VARCHAR(50),
    capacity INTEGER,
    yearOffered VARCHAR(20) NOT NULL,
    semesterOffered VARCHAR(20) NOT NULL
  );
`)
```



```
{
  "id": "CS-173",
  "cores": ["Q", "R"],
  "creditAmount": 4,
  "title": "Intro Comp Sci",
  "description": "CS-173. Introduction to",
  "yearOffered": "EVERY",
  "semesterOffered": "EVERY",
  "prerequisites": []
},
{
  "id": "CS-174",
  "cores": [],
  "creditAmount": 4,
  "title": "OO Programming",
  "description": "CS-174. Object-Oriented",
  "yearOffered": "EVERY",
  "semesterOffered": "EVERY",
  "prerequisites": ["CS-173"]
},
}
```

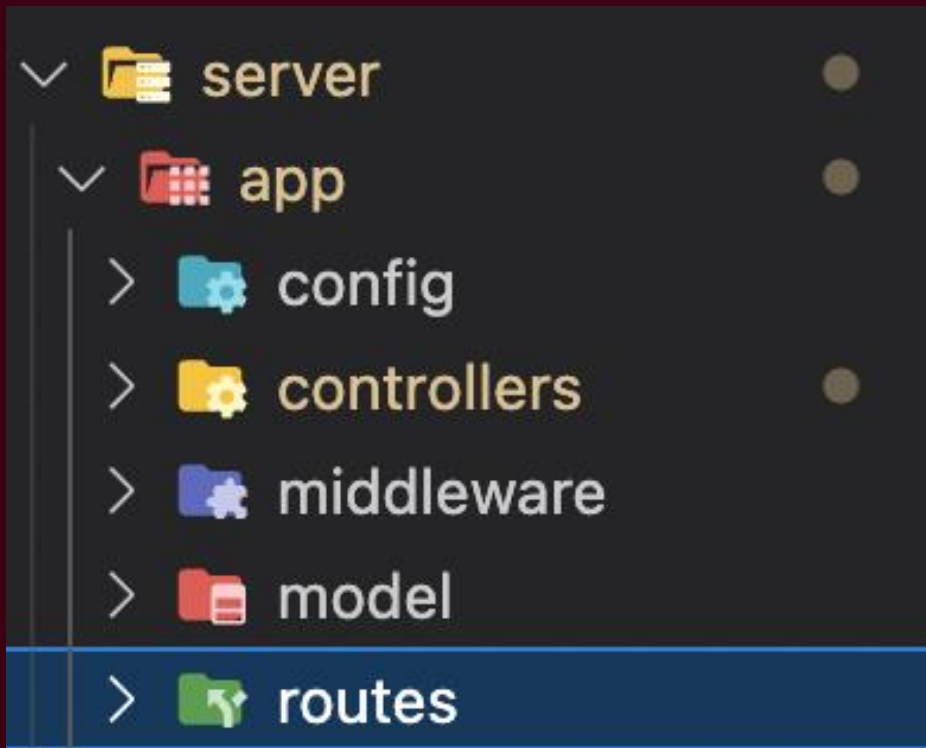


```
const addCourseToSemester = async (studentId, course, semester, fullPlan) => {
  const db = await fetchDB()
  const semesterDuplicates = Object.keys(fullPlan).filter(
    (key) => fullPlan[key].filter((courseEntry) => courseEntry.id === course).length > 0
  )
  if (semesterDuplicates.length > 0) {
    try {
      const query = 'INSERT INTO StudentCoursePlan (studentId, courseId, semester) VALUES (?, ?, ?)'
      const result = await db.run(query, [studentId, course, semester])
      console.log(`Inserted a row with the ID: ${result.lastID} - ${course} for ${semester}`)
    } catch (err) {
      throw new Error(err.message)
    }
  } else {
    throw new Error(`${course} is already planned for ${semesterDuplicates.join(',')}!`)
  }
}

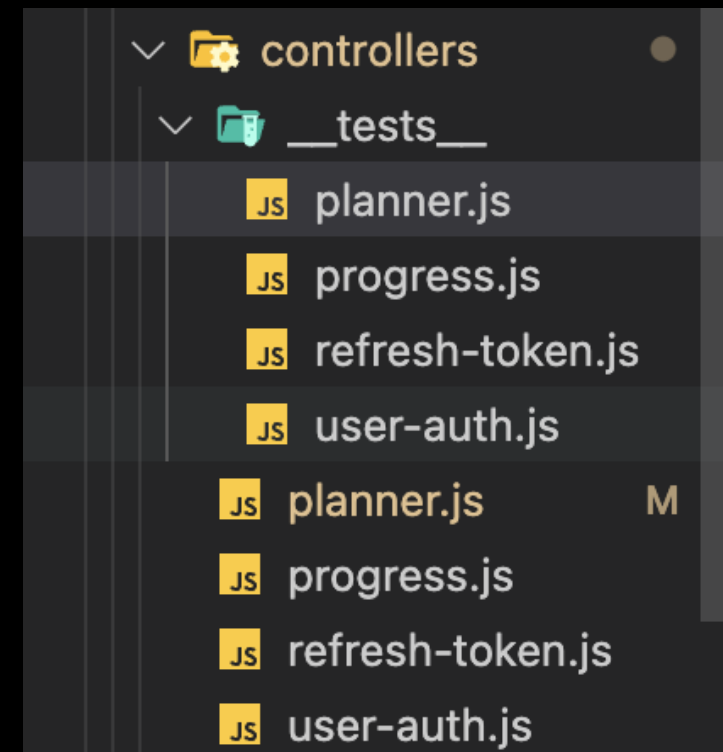
exports.addCourseToSemester = addCourseToSemester
```


Backend Structure

Backend Folder Structure



Test Structure



```

const verifyToken = (req, res, next) => {
  const token = req.headers['x-access-token']

  if (!token) {
    return res.status(403).json({ unauthorized: true })
  } else {
    jwt.verify(token, config.secret, (err, decoded) => {
      if (err) {
        console.log(err)
        return catchError(err, res)
      }
      req.userId = decoded.id
      next()
    })
  }
}

```

```

const { authJwt } = require('../middleware')
const controller = require('../controllers/planner')

module.exports = function (app) {
  app.use(function (req, res, next) {
    res.header(
      'Access-Control-Allow-Headers',
      'x-access-token, Origin, Content-Type, Accept'
    )
    next()
  })

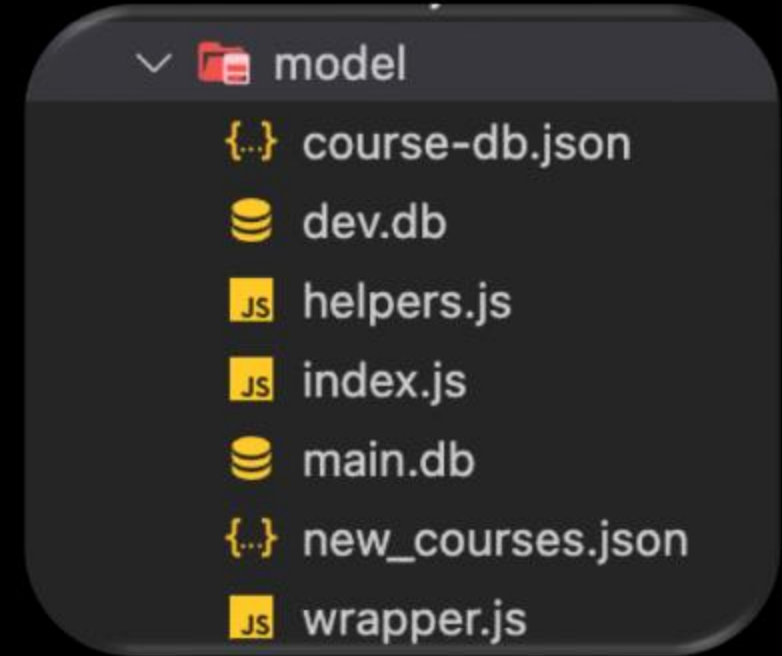
  // Adapted from: https://github.com/bezkoder/node-js-jwt-auth/blob/master,
  app.post('/removeCourse', [authJwt.verifyToken], controller.removeCourse)
  app.post('/addCourses', [authJwt.verifyToken], controller.addCourses)
  app.get('/fetchPlan', [authJwt.verifyToken], controller.fetchPlan)
}

```

Routes & Middleware

Testing Setup

- Separated in main and dev database
- Before each test, the dev database would be copied over
- The tests would be ran against the copied database ("dev_test.db")



```
beforeEach(async () => {  
  await setUpTestDB('dev.db', 'dev_test.db')  
})
```

Testing Example

```
exports.addCourses = async (req, res) => {
  const { courseIdList, semesterKey } = req.body
  const userId = req.userId
  const formerPlan = await getFullPlanFromDB(userId)
  if (Object.keys(formerPlan).includes(semesterKey) && formerPlan[semesterKey].length + courseIdList.length <= 4) {
    try {
      await Promise.all(courseIdList.map(async (courseId) => {
        await addCourseToSemester(userId, courseId, semesterKey, formerPlan)
      }))
      const newPlan = await getFullPlanFromDB(userId)
      res.status(200).json({
        fullPlan: newPlan,
        message: `Successfully added courses ${courseIdList.join(', ')} to ${semesterKey}!`
      })
    } catch (err) {
      res.status(500).json({ fullPlan: formerPlan, message: err })
    }
  } else {
    res.status(409).json({
      fullPlan: formerPlan,
      message: 'Only a maximum of four courses can be added to a semester\'s plan!'
    })
  }
}
```

```

it('should addCourses with status 200', async () => {
  const mReq = {
    userId: 1,
    body: {
      courseIdList: ['MATH-111', 'CS-174'],
      semesterKey: 'F2022'
    }
  }
  const mRes = {
    status: jest.fn().mockReturnThis(),
    json: jest.fn().mockReturnThis()
  }
  await plannerModule.addCourses(mReq, mRes)
  expect(mRes.status).toBeCalledWith(200)
  expect(mRes.json).toBeCalledWith(
    expect.objectContaining({
      message: 'Successfully added courses MATH-111, CS-174 to F2022!'
    })
  )
})

```

```

it('should addCourses with status 409 due to maximum course load', async () => {
  const mReq = {
    userId: 1,
    body: {
      courseIdList: ['MATH-111', 'CS-174', 'DANC-100', 'DANC-330', 'MATH-112'],
      semesterKey: 'S2023'
    }
  }
  const mRes = {
    status: jest.fn().mockReturnThis(),
    json: jest.fn().mockReturnThis()
  }
  await plannerModule.addCourses(mReq, mRes)
  expect(mRes.status).toBeCalledWith(409)
  expect(mRes.json).toBeCalledWith(
    expect.objectContaining({
      message: 'Only a maximum of four courses can be added to a semester\'s plan!'
    })
  )
})

```

Test Cases

```


1 name: Node.js CI
2
3 on:
4   push:
5     branches: [ "master" ]
6   pull_request:
7     branches: [ "master" ]
8
9 jobs:
10  build:
11
12    runs-on: ubuntu-latest
13
14    strategy:
15      matrix:
16        node-version: [16.x, 18.x]
17        # See supported Node.js release schedule at https://nodejs.org/en/about/releases/
18
19    steps:
20      - uses: actions/checkout@v3
21      - name: Use Node.js ${ matrix.node-version }
22        uses: actions/setup-node@v3
23        with:
24          node-version: ${ matrix.node-version }
25          cache: 'npm'
26      - run: npm ci
27      - run: npm run build_once
28      - run: npm test

```

All checks have passed

2 successful checks


✓  Node.js CI / build (16.x) (push) ... [Details](#)


✓  Node.js CI / build (18.x) (push) ... [Details](#)

✗ Carousel Fixes

Some checks were not successful

1 cancelled and 1 failing checks

ⓘ  Node.js CI / build (16.x) (push) Cancelled after 31s [Details](#)

✗  Node.js CI / build (18.x) (push) Failing after 25s [Details](#)

Github CI/CD