Oregon State University

CS462: Senior Software Engineering Project

A Web-Based Tool for Task-Delineated, AI-Assisted Assignments

Group 28:
Oliver Zhou, Ethan Lu, Trent Matsumura, Sai Anand, Collin Kimball

Oregon State University
College of Engineering

Project Introduction

This project is a website that acts as a more extensively featured Learning Management System, similar to Canvas.

Key Features:

- Task-delineation: Meaning divided tasks which enables complex structures for delivering assignments and tasks to students, allowing “stories.”
- AI-integration: Implements AI tools to help instructors develop class content and evaluate student performance, and help students use AI to improve their thought processes.

Oregon State University
College of Engineering

Problem Statement

Organization:

- Assignments are limited in their ability to represent complex structures in standard Learning Management Systems i.e. Canvas.

AI:

- Students are encouraged or incentivized to use AI tools to quickly finish assignments, which isn’t productive for improving mental fitness
- Punishing AI use does not stamp out the root of the problem, and misses the opportunity to use the technology for our advantage.

Oregon State University
College of Engineering

Requirements

Our Project Mentor, Sanjai, has laid out the key requirements for this project, and we aim to complete them according to his guidance/desires.

Basic Overview:

- Prototype a tool which functions similarly to Canvas, with these features:
    - Organized assignment structures
    - AI helper bots for students
    - AI report generation for instructors

Oregon State University
College of Engineering

Requirements (cont)

The scope of our requirements are flexible in scope, and our project mentor emphasized that things will change. We have already removed an old “collaborative” requirement.

Example functional requirements:

- Configurable AI models
- Dividable assignment tasks/stories
- Exportable assignment objects
- AI Chat Bot Window

Design

The frontend and backend of this website will be built mostly traditionally, using React, Express, and Node.js.

For the scope of our project, we decided to use a pre-existing technology for our AI implementation. The AI model we decided on was the Google chatbot model Gemini. We will integrate this AI model with API requests for our AI features.

Oregon State University
College of Engineering

Roadmap

Sprint 1: Initial front end layout, Preliminary testing of AI integration, Database testing

Sprint 2: Logging user chat history, User authentication, Database interaction with AI, AI functionality in a chatbot using Google Gemini.

Sprint 3: Further database creation with assignments, tasks, and chat logs, cleaned up the folders.

Sprint 4: Url linking assignments to chat bots with context, refined user roles.

Oregon State University
College of Engineering

Components

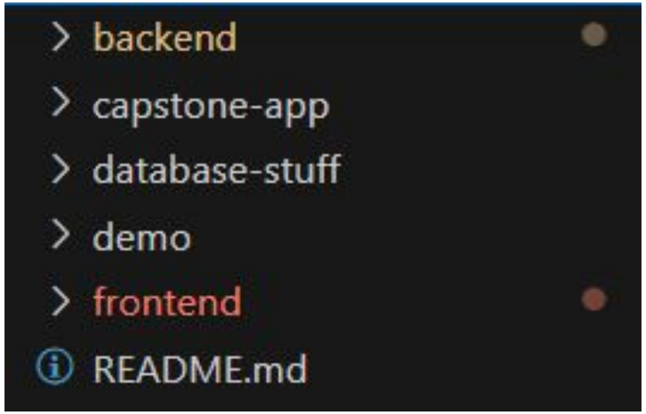
Core Components:

- Frontend
  - User Interface: Collects inputs and displays outputs
- Backend
  - Handles internal logic of the web-page
  - Includes part of API processing for AI components
- AI Components
  - Handles API requests with AI model

Oregon State University
College of Engineering

Code Architecture

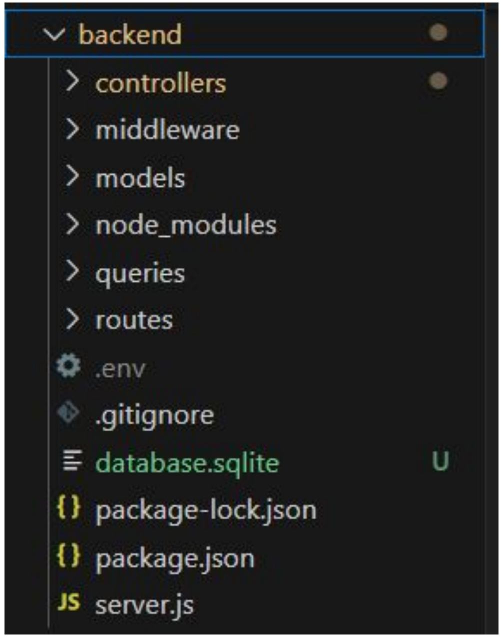
Our code architecture is currently set up like this, a folder for frontend work and backend work. The other three folder were used for the testing of the database, the prototype front end, and the demo used for the fall section of this class.


backend
capstone-app
database-stuff
demo
frontend
README.md

Oregon State University
College of Engineering

Code Architecture Cont

Furthermore our backend file is structured as the image shows. We have several folders showing the controllers, middleware, models, queries, and routes. We also included our env and gitignore file in this folder, as well as a database, and our server file.

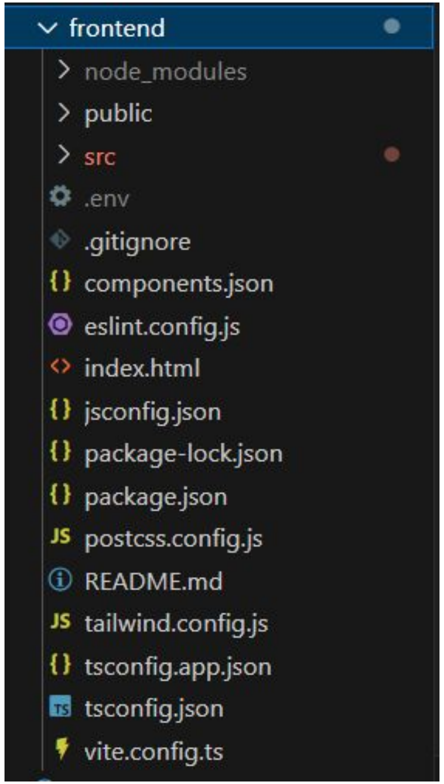


```
backend
  controllers
  middleware
  models
  node_modules
  queries
  routes
  .env
  .gitignore
  database.sqlite      U
  package-lock.json
  package.json
  server.js
```

Oregon State University
College of Engineering

Code Architecture Cont

The frontend file is structured as the image shows. Several folders, including the src folder which includes a majority of the work. It holds the components, api fetch methods, index styling file, and several other things. This folder also includes its own env and git ignore file.

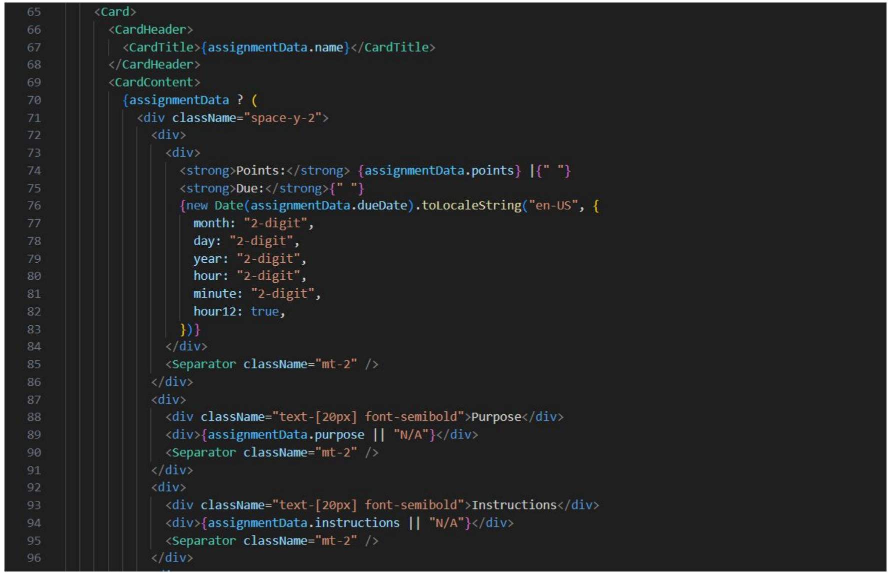


```
frontend
  node_modules
  public
  src
  .env
  .gitignore
  components.json
  eslint.config.js
  index.html
  jsconfig.json
  package-lock.json
  package.json
  postcss.config.js
  README.md
  tailwind.config.js
  tsconfig.app.json
  tsconfig.json
  vite.config.ts
```

Codebase

Example JSX page (AssignmentPage.jsx)

Part of the code involved within a single jsx page.



```
<Card>
  <CardHeader>
    <CardTitle>{assignmentData.name}</CardTitle>
  </CardHeader>
  <CardContent>
    {assignmentData ? (
      <div className="space-y-2">
        <div>
          <div>
            <strong>Points:</strong> {assignmentData.points} |{" "}
            <strong>Due:</strong>{" "}
            {new Date(assignmentData.dueDate).toLocaleString("en-US", {
              month: "2-digit",
              day: "2-digit",
              year: "2-digit",
              hour: "2-digit",
              minute: "2-digit",
              hour12: true,
            })}
          </div>
          <Separator className="mt-2" />
        </div>
        <div>
          <div className="text-[20px] font-semibold">Purpose</div>
          <div>{assignmentData.purpose || "N/A"}</div>
          <Separator className="mt-2" />
        </div>
        <div>
          <div className="text-[20px] font-semibold">Instructions</div>
          <div>{assignmentData.instructions || "N/A"}</div>
          <Separator className="mt-2" />
        </div>
```

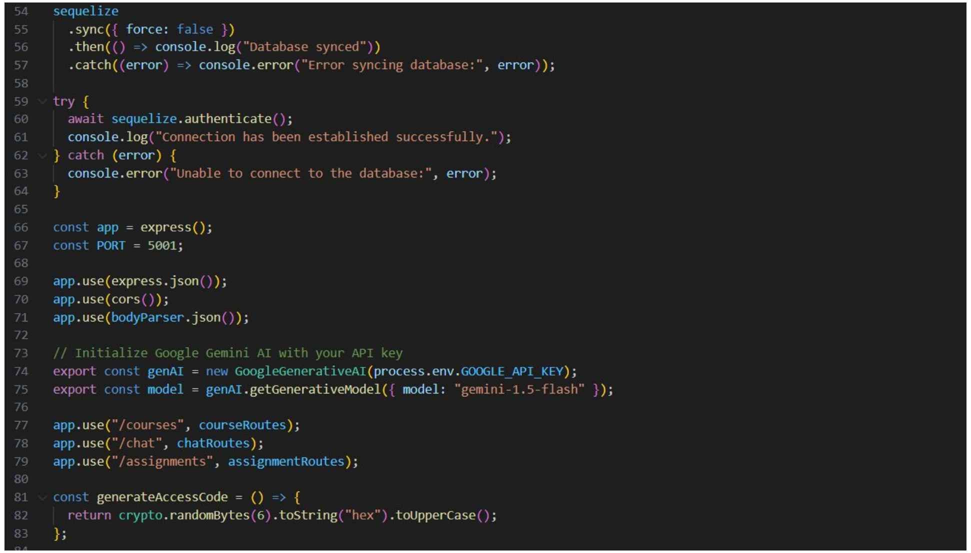
Oregon State University
College of Engineering

Codebase

Server.js

Handles creating the databases, starting the server, configuring the access codes.

A small portion of the code from that file.



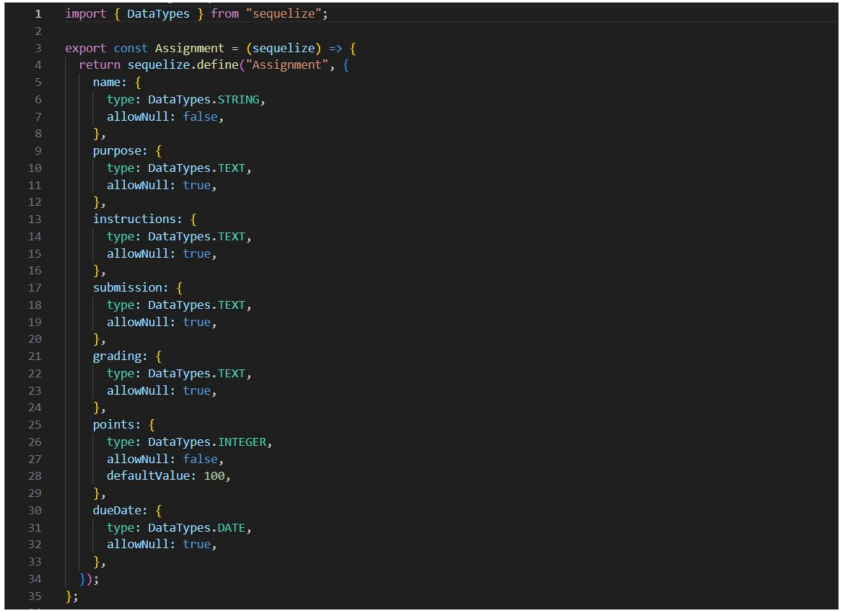
```
sequelize
  .sync({ force: false })
  .then(() => console.log("Database synced"))
  .catch((error) => console.error("Error syncing database:", error));

try {
  await sequelize.authenticate();
  console.log("Connection has been established successfully.");
} catch (error) {
  console.error("Unable to connect to the database:", error);
}

const app = express();
const PORT = 5001;

app.use(express.json());
app.use(cors());
app.use(bodyParser.json());

// Initialize Google Gemini AI with your API key
export const genAI = new GoogleGenerativeAI(process.env.GOOGLE_API_KEY);
export const model = genAI.getGenerativeModel({ model: "gemini-1.5-flash" });

app.use("/courses", courseRoutes);
app.use("/chat", chatRoutes);
app.use("/assignments", assignmentRoutes);

const generateAccessCode = () => {
  return crypto.randomBytes(6).toString("hex").toUpperCase();
};
```

Codebase

Models (assignment.js)

This is a page for the assignment.js. It contains the various variables that a single assignment would hold and is used to create the table for the database.

There are several more models, with each one serving a different purpose.



```
import { DataTypes } from "sequelize";

export const Assignment = (sequelize) => {
  return sequelize.define("Assignment", {
    name: {
      type: DataTypes.STRING,
      allowNull: false,
    },
    purpose: {
      type: DataTypes.TEXT,
      allowNull: true,
    },
    instructions: {
      type: DataTypes.TEXT,
      allowNull: true,
    },
    submission: {
      type: DataTypes.TEXT,
      allowNull: true,
    },
    grading: {
      type: DataTypes.TEXT,
      allowNull: true,
    },
    points: {
      type: DataTypes.INTEGER,
      allowNull: false,
      defaultValue: 100,
    },
    dueDate: {
      type: DataTypes.DATE,
      allowNull: true,
    },
  });
};
```

Oregon State University
College of Engineering

Software Demo

(What parts we did individually)

Next Steps

- Move our software to the OSU engineering servers rather than having to run it locally.

- Deploy our software to real world scenarios, allowing students and teachers to test and give feedback.

- Implementing additional quality of life features to our software, such as configurable chatbots and chatbot summaries of student interaction.

Conclusion

Feedback

We will continue to focus on implementing our project alongside our project mentor to make sure his concerns are addressed. We hope to address any bugs or other user issues that have been overlooked during the development process.

Results

Our software currently has the majority of the required functionality completed. However, several important secondary features still require implementation. These will be completed soon and we will plan to work further on new features after figuring out our migration to a server base from running it locally.

Oregon State University

Feedback?

Thank You!