# CS 4390

# Google Tech Exchange 2024

# Software Development Studio

# Syllabus

## 1. Course Information

### 1.1 Course Description

Software Development Studio integrates various components of students' undergraduate computer science curriculum, enabling students to work collaboratively on projects using professional tools and processes. This course bridges the gap between the academic and industry experience of software engineering.

### 1.2 Prerequisites

- Completion of two Computer Science courses at their university including Data Structures & Algorithms.
- Ability to solve coding problems using lists/arrays, dictionaries/maps, and graphs.
- Proficiency in a high-level programming language such as C, C++, Go, Java, JavaScript, Python, or Rust.

### 1.3 Class Schedule

| *SDS* | *Google Instructor* | Faculty Instructor | Course PgM |
|---|---|---|---|
| A (M/W 5-6:20pm ET) | Anna Prokofieva | Daniel Mejia* - UTEP | Charles Smith-De Ville |
| B (M/W 3-4:20pm ET) | Anna Prokofieva | Kianoush Gholamiboroujeni* - FIU | Pradeep Koka |
| C (T/Th 11-12:20pm ET) | Nicholas Kelman | Lei Qian* - Fisk | Jasmine Rogers |
| D (T/Th 3-4:20pm ET) | Nicholas Kelman | Jaycee Holmes - Spelman/Codehouse | Imani Herring |

### 1.4 Office Hours Schedule

All students are welcome to all office hours!

| *Day of the Week* | *Time* | *Hosts* |
|---|---|---|
| Tuesday | 5-6 PM ET | Nicholas Kelman |
| Thursday | 5-6 PM ET | Nicholas Kelman |
| Wednesday | 1-3 PM ET | Anna Prokofieva |

# 2. Course Objectives and Learning Outcomes

## 2.1 Course Objectives

This course aims to teach students:

- Understand the fundamentals of full-stack software development, including frontend, backend, and database architecture, and apply this knowledge to design and develop web applications.
- Develop effective debugging and troubleshooting skills to identify and resolve software issues, ensuring the reliability and quality of software applications.
- Apply testing strategies, including unit tests and integration tests, and employ software engineering practices, *such as Agile methodologies,* to ensure robust and high-quality software development.
- Gain proficiency in using version control systems, such as Git, to manage code changes and collaborate efficiently within a team.
- Explore cloud computing platforms/services and generative AI tools, and learn how to deploy, scale, and manage software applications securely and effectively.

## 2.2 Learning Outcomes

### Level 1

**The student has been exposed to the terms and concepts at a basic level** and can supply basic definitions. Upon successful completion of this course, students will be able to:

1. Define full-stack software development and its components, including frontend, backend, and database architecture.
2. Identify the key elements of software requirements and explain how they are used in the design and development process.
3. Explain the purpose and functionality of UI Design in web application development.
4. Recognize and describe the basic ethical considerations and principles relevant to computer science and software development.
5. *Understand software engineering practices, such as Agile methodologies, for effective project management and collaboration.*

### Level 2

**The student can apply the material in familiar situations**, e.g., can work a problem of familiar structure with minor changes in the details. Upon successful completion of this course, students will be able to:

1. Apply the principles of full-stack software development to design and build web applications, integrating frontend, backend, and database components.
2. Utilize cloud computing platforms and services to deploy, scale, and manage software applications, ensuring scalability, availability, and security.
3. Utilize version control systems, leveraging Github Classroom, to manage code changes, including branching and merge conflicts.
4. Implement testing strategies, including unit tests and integration tests, to ensure software reliability and quality.
5. Apply software engineering practices, such as Agile methodologies, for effective project management and collaboration.

### Level 3

**The student can apply the material in new situations**. This is the highest level of mastery. Upon successful completion of this course, students will be able to:

1. Analyze complex software requirements and design to develop robust and scalable full-stack web applications prototypes.
2. Collaborate with team members to integrate and manage shared codebases using version control systems, effectively resolving conflicts, and ensuring code consistency.
3. Integrate various APIs, frameworks, and libraries to develop advanced functionalities in software projects.
4. Communicate software designs effectively, both in written documentation and through verbal explanations, considering audience comprehension and technical accuracy.
5. Utilize code debugging techniques to identify and resolve issues in software.

# 3. Course Resources

## 3.1 Course Components

- **Lectures:** 1 hour 20 minutes, occurring 2 times per week.
- **In-class activities:** Group labs, discussion and in-class exercises during lectures.
    - Project Coaches will be available to help students work through in-class activities.
- **Group Project:** The main deliverable of the class, completed with a group of 3.
    - Project coaches will be available to assist and coach teams.
- **Individual Homework:** Practice for important skills taught in class.
    - Project coaches will be available to assist and coach individuals.

## 3.2 Course grade

| Assignment | # of Total Assignments | % of Grade | Notes |
|---|---|---|---|
| Attendance + Participation | 26 | 10% | Each class period students can receive up to 4 points total for attendance.<br>● **2 points** for camera on<br>● **1 point** for attending<br>● **1 point** for participating |
| Unit Assessments | 17 (3-4 per unit) | 40% | Each unit's homework is worth 8% of the final grade. |
| Final Project | 7 | 50% | 7 parts, each worth 3-10% of the final grade. |

The nominal percentage-score-to-letter-grade conversion is as follows:

- 90% or higher is an A
- 80-89% is a B
- 70-79% is a C

- 60-69% is a D
- below 60% is an F

The instructor reserves the right to adjust these criteria downward, e.g., so that 88% or higher represents an A, based on overall class performance. The criteria will not be adjusted upward, however.

## 3.3 Asking Questions

If students have questions outside of class, there are several venues where they can ask them:

- **Course email**:
  - If students have questions that they don't want to send publicly in the group chat (e.g. a question that includes a snippet of their code), they can email the course staff at tx24-sds-leads@techexchange.in
- **Office hours:**
  - Instructors will host office hours throughout the semester via video chat. You should see invitations to office hours on your Google Calendar. Feel free to attend office hours for any of the instructors, even if they aren't the instructor for your section.
- **Individually email your instructors or the Course PgM**
  - If you have questions specifically for your instructors or the Course PgM - all emails can be found within Google Classroom
- **Discord**
  - You should've been added to our Google Tech Exchange 24 Discord. In the different channels you can ask general questions, chat with instructors, TAs mentors,mock interviewers and other students
  - Channels: #general-techX → general TechX questions
  - Channels: #software-development-studio → Course questions, student collaboration, office hour reminders
- **SDS Project Coaches**
  - If you need additional assistance with your coding assignments you can also reach out to your assigned SDS Project Coach via email or discord.

## 3.4 Required Resources

None. Students will access assignments and readings via EdStem. Some useful resources that you may want to consult are:

- Cloud Shell: cloud.google.com/shell
  - Google Vertex API
  - Google Makersuite API
- Git: lab.github.com
- Python: docs.python.org
- Streamlit: blog.streamlit.io
- Figma: help.figma.com

# 4. Units Covered

| Unit | Topics |
|---|---|
| 1 | Introduction + Setting up your environment |
| 2 | Project Planning + System Fundamentals |
| 3 | Building Application: Cloud Tools |
| 4 | Building Application: Frontend Tools |
| 5 | Deploying Applications |
| Final Project | Comprehensive Project of all Tools learned |

# 5. General Policies & Academic Integrity

## 5.1 Attendance Policy

Students are expected to attend classes regularly, on time, and **with cameras on**.

- You must attend classes on time and actively participate during group work to get credit. If you have to miss class, please reach out to the course instructors.
- All students will be required to have cameras on to receive full attendance credit.
- There will be brief quizzes and surveys during some classes. These quizzes will be graded for completion, NOT for correctness. The quizzes are intended to provide feedback to your instructors.
  For each class period, you can receive a total of 4 points:
  - **2 points** for camera on
  - **1 point** for attending
  - **1 point** for participating

If you need an extension or have to miss class, please submit this form.

## 5.3 Late Work policy

Submission of late work is subject to approval by your Project Coach (your project coach must agree to grade your late work). Late assignments or late implementation of individual requirements will be **penalized 50% (granted points divided by 2)**. Each assignment can only be submitted once.

- Example: Student submits late assignment that meets 85/100 points. Grade given is 42.5/100.

## 5.4 Extensions

Extensions are only allowed through Tech Exchange (i.e. in emergency situations). If you feel like you will not complete the assignment on time, talk with your project coach and instructor to get increased support instead of lengthened deadlines. If you need to request an extension, submit this form.

## 5.5 Plagiarism policy

Use the table below to determine if you can collaborate on a given assignment.

### What kind of help can I get on this assignment?

| Type of work | Help allowed from |
|---|---|
| **In-class activity** | Instructors, project coaches, or student enrolled in this class |
| **Final Group Project** | Instructors, project coaches, or students on the same team. |
| **Individual Homework Assignments** | Instructors or project coaches. Further collaboration permitted will be detailed in homework assignments. |

### Guidelines for Collaboration

Software engineering is inherently a collaborative process and in this class, we expect students to collaborate with **only** their fellow student(s) they are paired with in labs and grouped into for the unit projects.

All work submitted for this assignment must be your own. You may use generative AI tools like ChatGPT or Bard to help you complete this assignment, but all code from these tools **must be clearly marked** (explicitly comment the start line and end line of generative AI code). Failure to mark generative AI code could result in your code getting flagged for plagiarism. Any instances of plagiarism will be dealt with according to your university's academic integrity policy.

## 5.6 ADA Policies and Procedures

If a student needs particular accommodations to be made, they must submit this form. Tech Exchange and your instructor will then work with you to make sure we are meeting the accommodations set by your institution.

# 6. Using Generative AI to Write Code

## 6.1 Policy & Guidelines

You are allowed to use Generative AI Tools like Bard and ChatGPT in whatever way you want (or not at all). However, you must conform to these rules:

1. **Mark all code snippets copied from a generative AI tool.** This is the only way to ensure that your code is not flagged for plagiarism. See 6.3 Generative AI Code Example.
2. **Fill out the Generative AI TechExchange surveys honestly.** We are allowing the use of these tools as part of Tech Exchange's research on how students use generative AI when given the option. We want to know how it is helpful for students, as well as how it is unhelpful.

Remember these guidelines when using Bard or ChatGPT.

- Do not spend more than 1 hour trying to fine-tune a generative AI prompt. ***You are a better engineer than Bard or ChatGPT!***
- While you are allowed to submit code that is 100% generated, always run your code before submitting to verify that it works as intended. No matter how small the error, ***if your code does not run, you will not get points.***
    - Additionally, if your code ***does not meet the assignment requirements***, you will not get points.

## 6.2 Tips for Prompts

Generative AI is most helpful for these uses:

- Coming up with a general idea related to a specific interest (e.g. “give me an idea for a streamlit app related to baseball”)
- Creating a template
- Generating starter code
- Understanding how to code a specific small task (less than 20 lines of python code)
- Generating sample data

Generative AI responses are often **confidently wrong**. You may ask something like “what is a unit test” and you will receive a long, smart-sounding answer. The answer is probably correct, but it will have missing pieces and might contain contradictory information. In this case, you will get better results by running a google search and reading a real article on the topic.

Generative AI is usually **NOT** helpful in these cases:

- Writing tests
- Explaining concepts (doing a Google search a clicking on a specific result is better)
- Finding a bug in your code
- Generating more than ~50 lines of code

## 6.3 Generative AI Code Examples

### Code Snippet Example

```
import streamlit as st
import backend

data = backend.load_data()

# START OF BARD CODE
# Create a sidebar for selecting the data to display
st.sidebar.header("Select data to display")
selected_measure = st.sidebar.selectbox("Data type", ("Temperature", "Sea level", "Precipitation"))
# END OF BARD CODE

st.line_chart(data, x="Year", y=selected_measure)
st.table(data)
```

## Starter Code Example

```
# Used Bard to create a hello world starter template.
# Wrote my own function to display data from my backend.
import streamlit as st
import backend

# Title and header
st.title("Hello, world!")
st.header("Welcome to your first Streamlit app!")

def display_table():
  data = backend.load_my_data()
  st.table(data)

button = st.button("Generate Data")
if button:
  display_table()

# Run the app
if __name__ == "__main__":
  st.sidebar.title("About")
```