07-15 Lecture: Core Principles and Team-Based Assignments in Web Application Development

Date & Time: 2025-07-15 13:34:36
Location: BSP (likely a classroom setting at UBC)
Title: Course Introduction: Web Application Development Fundamentals
`web application development` `team-based assignments` `academic integrity`

Theme

This lecture, delivered by Karthik, outlines a fast-paced web application development course taught at BSP, emphasizing core principles over specific technologies. Key points include team-based cumulative assignments, strict deadlines, academic integrity, and the exclusive use of private GitHub repositories for submissions. The grading scheme, participation policies, and exam structure are detailed, with a strong focus on student responsibility and collaboration.

Takeaways

1. Core principles and abstractions in web application development are more important than specific technologies, as technologies change rapidly and principles remain relevant.
2. The course is fast-paced and covers material in approximately three and a half weeks.
3. Assignments are team-based (teams of three) and constitute the bulk of the grade.
4. There are four assignments, each cumulatively building on the previous one, meaning missing one prevents completion of subsequent ones.
5. Assignments account for 60% of the grade; the final exam is 40%.
6. No late assignments are accepted, with a very low threshold for exceptions, and deadlines are hard (11:59 PM, with an automated script running at 12:01 AM).
7. Assignment submission is exclusively via private GitHub repositories.
8. Class activities require laptops or tablets; phones are not recommended for programming tasks.
9. No textbook is required; lecture notes, code, and resources are provided online via a course website and Git repository.

10. Attendance is not mandatory but highly recommended due to the fast pace and dedicated in-class time for activities and assignment work.

Highlights

- `"If you learn the core techniques, the core abstractions, you can always pick up the technology."-- Speaker 1`
- `"Don't cheat yourself out of an education by doing all this."-- Speaker 1`
- `"I do want you to enjoy the class till it's impossible, but with that said, I also want to make sure, you know, you understand your responsibilities, right, so I'll treat you all as adults, because most of you are, I guess, and you need to assume responsibility for your actions."-- Speaker 1`

Chapters & Topics

Core Principles Over Technology

> The course emphasizes understanding core abstractions and principles in web application development rather than focusing on specific, potentially outdated technologies.

- **Keypoints**
  - Technologies change rapidly, but core principles remain relevant.
  - Learning principles enables adaptation to new technologies.
  - Course content is designed to be technology-agnostic, encouraging students to build their own libraries based on principles.
- **Explanation**
  The instructor, Karthik, who has been at UBC since 2010 and first taught a version of this course in 2014-2015, highlights that while technologies have changed considerably, the core abstractions and principles have not. Students are encouraged to focus on learning these fundamentals to future-proof their skills, rather than just copying code or relying on specific frameworks.
- **Considerations**
  - Do not focus solely on learning the latest frameworks or libraries.
  - Be prepared to develop your own libraries based on the principles taught.

Team-Based Assignments and Grading

> Assignments are completed in teams of three, with all members receiving the same grade unless a member cannot demonstrate understanding of the work or has not contributed.

- **Keypoints**
  - There are four assignments, each cumulatively building on the previous one.
  - Assignments account for 60% of the total course grade, making them the bulk of the assessment.
  - All team members must actively participate; non-participation or inability to answer questions about the work can result in a zero for that assignment.
- **Explanation**
  Assignments are cumulative, requiring completion of previous ones to proceed. If a team member is found not to have contributed or cannot answer questions about the assignment, they may receive a zero, even if other team members completed the work. This policy encourages equitable participation and skill development within teams.
- **Considerations**
  - Collaborate effectively and equitably within your team.
  - Notify the instructor if a team member is not contributing to ensure fair assessment.
- **Special Circumstances**
  - If a team member does not participate, inform the instructor; they may be questioned and could receive a zero for the assignment.

Assignment Submission and Deadlines

> Assignments are submitted via private GitHub repositories and are typically due every 2 to 3 days at 11:59 pm. Deadlines are hard, and no late submissions are accepted, with exceptions only for truly extenuating circumstances and a very low threshold.

- **Keypoints**
  - Assignments are not accepted late; exceptions are extremely rare and require truly extenuating circumstances.
  - Assignments are cumulative; missing one prevents completion of subsequent ones, as there are no sample solutions provided.
  - The submission script runs automatically at 12:01 AM, so code must be pushed to the designated private repository branch by 11:59 PM on the due date.
  - Only remote pushes are accepted; local commits are not sufficient for submission.
- **Explanation**
  Students must submit assignments on time via their private GitHub repositories. If

an assignment is missed, students cannot proceed to the next, as each builds on the previous, and no sample solutions are provided to help catch up. The strict deadline is enforced by an automated script that pulls submissions precisely at 12:01 AM.

- **Considerations**
  - Plan your work to meet tight deadlines and push code well before 11:59 PM to avoid missing the automated pull.
  - Contact the instructor immediately in case of truly extenuating circumstances, but be aware of the very low threshold for exceptions.
- **Special Circumstances**
  - If truly extenuating circumstances prevent submission, contact the instructor to discuss possible accommodations, though exceptions are rare and not guaranteed.

Final Exam Structure

> The final exam is worth 40% of the grade and consists of two parts: a 15% multiple choice section (closed book) and a programming section (open book/notes) with automated test cases.

- **Keypoints**
  - The multiple choice section is closed book and closed board.
  - The programming section is open book/notes and uses an online platform for problem-solving.
  - Passing all provided test cases is required for full marks on programming problems; there is no partial credit for partially working code.
- **Explanation**
  The final exam is designed to reflect class activities and assignments. Programming problems must pass all provided test cases for credit, and hardcoding solutions for specific test cases is not allowed. Students will know their results immediately upon passing the test cases.
- **Considerations**
  - Prepare thoroughly for both theoretical (multiple choice) and practical (programming) components.
  - Focus on robust solutions that pass all test cases, rather than attempting to hardcode for specific inputs.

Academic Integrity and Use of Gen AI

> Plagiarism and copying between teams is strictly prohibited. Use of Generative AI (e.g., ChatGPT) is not allowed for completing assignments or the final exam.

- **Keypoints**

    - Assignments must be original and completed by the team members themselves.
    - Generative AI tools can be used for understanding material (e.g., explaining concepts), but not for generating solutions for assignments or exams.
    - Violations of academic integrity, including unauthorized use of Gen AI for graded work, will result in an F for the course.
- **Explanation**
  Students are explicitly warned against using Gen AI for assignment or exam completion. If Gen AI is used for learning, students must verify the answers it provides, as AI can hallucinate or provide incorrect information. Relying solely on AI-generated answers is not a valid excuse for incorrect work.
- **Considerations**
    - Always verify any information obtained from Generative AI tools against reliable sources or by consulting the instructor.
    - Do not submit work that has been generated by AI.
- **Special Circumstances**
    - If unsure about the validity of AI-generated information or how to use AI ethically for learning, consult the instructor for clarification.

Course Logistics and Participation

> The course is fast-paced, with lectures, interactive activities, and dedicated assignment work time delivered over approximately three and a half weeks. Attendance is not mandatory but highly recommended to keep up with the rapid pace and benefit from in-class work.

- **Keypoints**
    - Lectures are interactive, include in-class activities, and are delivered by Karthik, with support from TAs Abraham (machine learning reliability, robust machine learning) and Mohsen (security, Internet of Things, real-time embedded devices).
    - Laptops or tablets are required for participation in activities and assignment work; phones are not suitable for programming tasks.
    - Lecture notes, code, and resources are provided online via a course website and Git repository, which are accessible to all students.
    - A Slack channel is used for communication, linked from the course website.
- **Explanation**
  Students are strongly encouraged to attend and participate in class to keep up with the rapid pace and utilize the dedicated time for activities and assignments. Missing classes can make it very difficult to catch up, as the course moves quickly and all exam material is drawn from lectures. The instructor is available after class for questions.

- **Considerations**
  - Bring a fully charged laptop or tablet to each class for activities and assignment work.
  - Actively use the course website and Slack channel for communication, resources, and staying updated.
  - If you miss a class, it is your responsibility to catch up with peers or via Slack.
- **Special Circumstances**
  - If you do not have a laptop or tablet, speak to the instructor for assistance.

Minimum Passing Grade

> The minimum passing grade for the course is 50%.

- **Keypoints**
  - Students must achieve at least 50% to pass the course.
  - Both assignments and the final exam contribute to the final grade, allowing multiple avenues to reach the passing threshold.
- **Explanation**
  This minimum passing grade was clarified in response to a student question. Students can pass the course by performing well in either the assignments or the final exam, or a combination of both.
- **Considerations**
  - Students should aim to perform well in both assignments and the final exam for a strong overall grade.

UBC Grading Scheme

> The grading scheme used in the course is similar to the UBC grading scheme. An A is 85%, an A+ is 90%, and 50% is a pass. There is a range of grades between these points.

- **Keypoints**
  - A is 85%
  - A+ is 90%
  - 50% is a pass
  - Full grading details, including the entire gamut of grades, can be found online via the UBC grading scheme.
- **Considerations**
  - Students should familiarize themselves with the detailed grading scheme to understand grade expectations and how their performance translates into letter grades.

Assignment Submission via GitHub

> Assignments must be submitted through private GitHub repositories created for each group. Each assignment requires a separate branch (e.g., `assignment1`, `assignment2`), and working code must be pushed to this branch before the hard deadline of 11:59 PM, as the automated script runs at 12:01 AM.

- **Keypoints**
  - A private GitHub repository is created for each group by the TAs after group formation.
  - Groups must be formed and member names submitted via a Google form by the end of the first class.
  - An assignment branch must be created for each assignment (e.g., `assignment1`, `assignment2`, `assignment3`, `assignment4`).
  - Working code must be pushed to the designated branch by 11:59 PM on the due date; the automated script pulls at 12:01 AM.
  - Only remote pushes are accepted; local commits are not sufficient for submission.
  - Students are strongly encouraged to commit code frequently during development as a good version control practice.
- **Explanation**
  Students form groups and submit their names via a Google form. TAs then create private repositories and add all group members as collaborators. For each assignment, students must create a specific branch and ensure their working code is pushed to this branch before the strict deadline. The automated submission process pulls code precisely at 12:01 AM from the branch, making late pushes unacceptable.
- **Considerations**
  - Commit and push code well before the 11:59 PM deadline to ensure it's captured by the automated script.
  - Familiarize yourself with Git branching and pushing procedures.
  - Do not attempt to commit to the public course repository for assignment submissions.
- **Special Circumstances**
  - If encountering truly extenuating circumstances that prevent timely submission, contact the instructor, but be aware that exceptions are extremely rare.

Class Participation and Bonus Points

> Participation in class and on Slack is encouraged. Bonus points are awarded for active participation, such as asking or answering questions. Participation is not mandatory, and lack of it does not penalize students.

- **Keypoints**

    - Bonus points are awarded for significant class and Slack participation (e.g., asking good questions, answering classmates’ questions).
    - The system was changed from a mandatory 5% participation grade to a bonus system due to student feedback.
    - There is no penalty for not participating, ensuring students who are less comfortable speaking up are not disadvantaged.
- **Explanation**
  Students are encouraged to participate in class discussions and activities, as well as on Slack. Those who contribute significantly may receive bonus points, which can help improve their final grade. This system aims to incentivize participation for the benefit of all students without penalizing those who prefer not to.
- **Considerations**
    - Participation helps deepen learning and benefits all students through shared understanding.
    - Bonus points can positively impact final grades.

Instructor Availability and Student Responsibility

> The instructor is available after class for at least 15-20 minutes to answer questions and help with exercises. Students are expected to take responsibility for their learning, keep up with the fast-paced course, and act with academic integrity.

- **Keypoints**
    - The instructor remains available after each class (typically 1:30 PM to 2:45 PM) for at least 15-20 minutes to assist students.
    - Students should use this time to complete exercises, clarify doubts about lecture material, and ask questions.
    - Students are responsible for their own actions, including keeping up with the course pace, seeking help when struggling, and maintaining academic integrity (e.g., avoiding plagiarism).
- **Explanation**
  The instructor encourages students to utilize the dedicated time after class for direct support and clarification. He emphasizes that students are treated as adults and are expected to assume responsibility for their learning journey, including proactively seeking help if they fall behind or struggle with concepts, and adhering to all academic policies. The course moves quickly after an initial slower period.
- **Considerations**
    - Take initiative to seek help from the instructor if struggling or needing clarification.
    - Maintain strict academic integrity in all submitted work.

Group Formation and Collaboration

> Students must form groups (typically of three members) and submit their group member names via a provided Google form by the end of the first class. Each group will then be provided with a private GitHub repository for their assignments.

- **Keypoints**
  - Groups of three are typical for assignments.
  - Group member names must be submitted via the Google form by the end of the first class to facilitate repository creation.
  - Each group will receive a private GitHub repository created by the TAs, with all members added as collaborators.
- **Explanation**
  The instructor emphasizes the importance of forming groups early to facilitate the setup of private GitHub repositories, which are essential for assignment submission. Working in teams is considered an important skill in software development, and students are encouraged to cultivate effective collaboration within their groups.
- **Considerations**
  - Proactively find group members and submit the Google form promptly.
  - Ensure all group members can access and work from the private repository.
  - Cultivate effective teamwork and collaboration within your group, as this is an important skill.

Assignments & Suggestions

- Complete four cumulative assignments in teams of three, each building on the previous one, and submit them via private GitHub repositories.
- Assignments are due every 2 to 3 days at 11:59 pm, with code needing to be pushed to the designated assignment branch before this hard deadline (the automated script runs at 12:01 AM).
- Participate in in-class activities using a laptop or tablet.
- Join the course Slack channel for communication and use the course website and Git repository for lecture notes and resources.
- Do not use Generative AI (e.g., ChatGPT) for assignments or the final exam; always verify any information obtained from AI.
- Prepare for a final exam consisting of multiple choice (closed book) and programming problems (open book/notes) where all test cases must pass for full credit.
- Form groups (typically of three) and submit group member names via the provided Google form by the end of the first class.
- Utilize the private GitHub repository created for your group (TAs will assist) for all assignment submissions, ensuring code is pushed to the correct branch.
- Participate in class and on Slack for bonus points.

- Introduce yourself in class, including your university, country, and a non-programming hobby.
- Utilize the instructor’s availability after class for questions and help with exercises.
- Take responsibility for keeping up with the fast-paced course and maintaining academic integrity.