

# CS100's & Beyond

Things to think about as you go through your CS degree to set yourself up for success



Michael Gathara,  
Samantha Miller,  
Christian Collins,  
Rizwan Khan  
& ACM @ UAB

# Hemingway's Law of Motion

Change happens gradually and then suddenly

**“Things happens slowly then all at once”**

You will start your degree and finish it before you even realize. Each year brings new opportunities, try to seize them before they are gone.

# Overview

- CS Clubs
  - UAB CS Clubs & Broader CS Clubs
- UAB CS Curriculum
  - BA vs BS and ABM
- Success in Class
  - How to be successful in your classes
- Personal Projects
  - The importance of personal projects
  - How to get started on personal projects
- Git & Github
  - Getting Started with Git and Github
- Internships and Grad School
  - Setting yourself up for internships
  - Setting yourself up for Grad School
- Plagiarism
  - Why to not do it
- AI Tool Usage to Enhance Learning

# CS Clubs

Joining a CS club is an important part of your CS degree. CS clubs expose you to new projects, ideas, allow you to build a community and make your CS experience a lot easier.

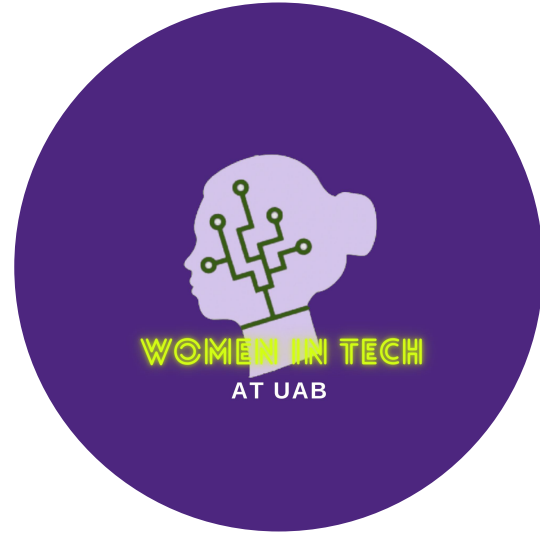
The next two slides showcases the CS clubs you can join here at UAB as well as nationally. At a minimum, we recommend joining both UAB ACM and UAB WIT and attending as many of their events as you can!

CS Clubs @ UAB | Both open to all students



ACM

[uabacm.org](http://uabacm.org)



WIT

[uab\\_womenintech](http://uab_womenintech)

# Broader CS Clubs

- Broadly, CS is filled with national clubs
- [ACM](#) - Association for Computing Machinery | Research focus
- [WIT](#) - Women in Tech Global | Community focus
- [Colorstack](#) - Black and LatinX students | Job and community focus
- [BIT](#) - Black in Tech | Community focus
- And many more -
  - [Clubs Advancing Diversity](#)
  - [Google Developer Clubs](#)

# UAB CS Curriculum

BA vs BSc and ABM



# A word on BS vs BA

UAB CS offers two Bachelors programs:

It also offers two bachelors with other departments such as Bioinformatics and Digital Forensics

- Bachelor of Science in CS

- Requires more Math and CS classes compared to BACS
- Offers automatic acceptances into the MSCS and MSDS if > 3.0 GPA
- Fulfills all requirements for the Accelerated Bachelors Masters (ABM) program

- Bachelor of Arts in CS

- Offers automatic acceptance into the MS in CyberSec if > 3.0 GPA
- You will have to take extra classes if you choose to enroll in the Accelerated Bachelors Masters (ABM) program and pursue the MSCS and MSDS track.
  - MSCS = CS 332, CS 350, CS 401
  - MSDS = MA 126, MA 260, CS 355



# ABM

UAB CS also offers the ABM Program:

- Accelerated Bachelors Masters
  - Allows you to share 12 hours of credit between your Bachelors and Masters. Also allows you to pay undergrad pricing for grad classes
  - Great option if you would like to acquire a Masters soon after Bachelors graduation
  - Enroll after finishing the core 300 level classes.

The ABM is for exceptional students. Acceptance into the program is typically at the end of junior year. Admission requirements include:

- a minimum of 75 credit hours (36 of these credit hours must have been taken at UAB)
- 3.5 (or higher) GPA in courses required by the student's undergraduate CS degree
- 3.5 (or higher ) overall UAB GPA
- completion of the BS/BA course requirements, except for 400-level courses
- completion of any necessary additional work (see the "Pairing of Programs" section)
- recommendation of acceptance by the admissions committee

Application deadlines are as follows:

- **Fall Admission (first M.S. courses would be taken in Fall):** July 1 for all MS programs
- **Spring Admission:** November 1 for all MS programs
- **Summer Admission:** March 1 for M.S. in Cyber-Security (not available for MSCS/MSDS)

Find out more [here](#)

# Success in Class



# Mastering Assignments

- **Understand Policies on Resources:**
  - Know what resources are allowed for assignments. When in doubt, always ask the professor or a TA!
- **Ask Questions:**
  - Don't be afraid to ask questions if you are unsure or curious to learn more.
- **Practice Problem-Solving Skills:**
  - Break large problems down into smaller sub-problems, use a pen & paper to write things out, work backwards, etc.
- **Write Clean Code:**
  - Keep your code clean, readable, and well-commented. Adhere to class standards.
- **Practice & Practice Some More:**
  - Practicing is vital for building skill, familiarity, and confidence.

# Pro tips for Labs & Homework

- **Labs:**

- **Take Advantage of Lab Time:**
  - Use lab time to work on assignments and seek guidance from TAs when needed.
- **Collaborate When Allowed:**
  - When allowed, working on lab assignments with others can be hugely beneficial.
- **Understand Key Concepts:**
  - Labs are designed to build up to larger assignments like homework, projects or exams. Make sure you understand the concepts behind each lab assignment and seek guidance from TAs and/or the professor when needed.

- **Homework:**

- **Start Early:**
  - Ideally, begin homework assignments as soon as they are released. Avoid procrastinating. Many assignments will take much longer than you might think at first glance.
- **Create an action plan:**
  - Try to create an action plan for each homework to ensure you fully understand the assignment prior to starting. This lets you clarify the assignment with the TAs before its too late
- **Understand the Assignment:**
  - Make sure you fully understand the assignment and what is being asked.
- **Break it down:**
  - Break large assignments down into smaller, manageable parts and tackle them one at a time.

# ACM Quizlets



[acmatuab.org/quizlets/](https://acmatuab.org/quizlets/)

Find helpful quizlets for many core CS classes on the ACM  
website

# Personal Projects



# Why do personal projects

Doing personal projects is often the best way to enhance and apply the things you learn in class. Other than the fact that it will help you showcase your skills. It offers several benefits

- Skill Development
  - Reinforces the skills you learn in class and allows you to practice building things from start to finish
  - Exposes you to more than you'll get a chance to learn in class. You can only cover so much in class.
- Portfolio Building
  - Allows you to demonstrate your technical expertise to potential employers
  - Highlights your coding abilities as often you can not post classwork to public repositories
- Find out what you like
  - Easy way to find out what kind of jobs or things you like to work on. Whether it be web development or systems or programming language development, etc.
  - Finding out what you like in particular can help increase motivation and confidence in CS

# Starting Projects

The best way to start on personal projects is by starting small with projects that allow you to practice what you are learning in class.

- Freshman Year: Build Foundations
  - Focus on learning the fundamentals and Python
  - Make basic programs like calculators, guessing games, to-do apps
  - Learn the basics of Git and Github (next section)
  - Join school CS clubs to meet more people and work with with them on projects
- Sophomore Year: Experiment and Showcase
  - Focus on multi-file projects using classes, Python or Java
  - Start digging into Web Development -
    - HTML/CSS/JS and build a small portfolio website
    - NodeJS, FastAPI, Flask for backend
  - Start practicing on Leetcode using [Leetcode Easy's](#)
  - Start utilizing Github for every personal project you do
  - Follow tutorials online to learn more about best practices

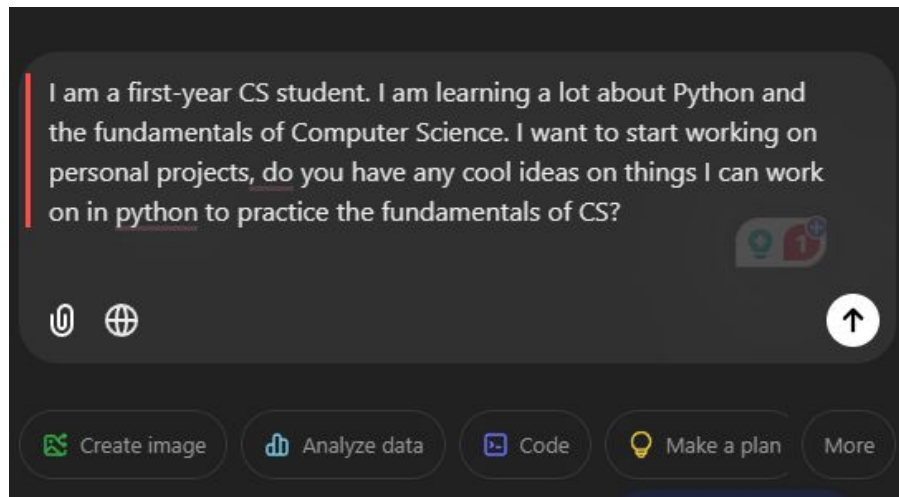


# Starting Projects Pt2

- Junior Year: Enhance and Build Portfolio
  - Start specializing, explore the areas of CS that seem interesting to you. AI/ML, Cybersecurity, Cloud Computing, Web Development, etc
  - Build mini-capstone level projects - projects that take weeks to complete
  - Enhance your online presence - do hackathons, ensure your Github profile has several repositories, start looking into [open source contributions](#)
  - Work on at least two group personal projects
  - Seek internships (next next section)
- Senior Year: Finalize and Polish
  - Collaborate with your group mates to ensure your capstone project is as best as can be
  - Have a polished portfolio website with several examples of your work over the years
  - Start applying to new grad roles, use the career center for resume reviews, career fairs
  - Attempt to have a few PRs opened/merged on open-source projects
  - Handle corner-cases on Git, such as merge conflicts, rebasing, etc

# Starting Projects Pt4

- A great way to get ideas on what to do as a personal project is by using ChatGPT. Try the prompt:
- I am a **XXXX**-year CS student. I am learning a lot about **XXXX** and **XXXX**. I want to start working on personal projects, do you have any cool ideas on things I can work on in **XXXX** to practice the **XXXX**



Fill out the **XXXX** with your information, like above

# Git & Github

Utilizing Git and Github



# Git **vs** Github/Gitlab/BitBucket

## Git:

- Source/Version Control
- Tracks changes made to code
- Works locally
- The underlying technology for Github, Gitlab, Code-Commit, and Bitbucket
- Learn it once, Use it everywhere
- Can be self-hosted to create your very own Github

## Github/Gitlab/BitBucket:

- Cloud-based hosting service that lets you manage repositories
- Utilizes Git as it's underlying technology, hence "Git as a service"
- Graphical user interface to view files
- No need to know Git to use
- Super popular

# Why Source/Version Control?

- Industry Standard
- Keep track of changes within files
- Create checkpoints as you work on a problem
- See your progress through time
- Access your files anywhere in the world
- Allow multiple people to work on the same project without affecting one another's work

# Installing Git

- Check if you have Git installed
  - In your terminal, run the command `git version`
  - The output will either tell you which version of Git is installed, or it will alert you that `git` is an unknown command.
- Git can be downloaded from the following link
  - <https://git-scm.com/downloads>
- And here are the steps for the installation process
  - <https://github.com/git-guides/install-git>

# Keywords

- **Repository**

- Similar to a directory, stores everything related to your project including files, versions, commits, etc.

- **Clone**

- Creates a linked copy of a repository that will sync with the original.

- **Fork**

- Creates an independent copy of a repository.

- **Branch**

- A version of the repository that allows one to test changes without testing the main repository

# Keywords

- Add

- When making changes, using add will store the file on a staging area where it will wait to be included in the next commit.

- Commit

- A 'snapshot' of changes made to the file, branch, or repository.

- Push

- Updates the remote repository with the commit.

- Origin

- The primary or original version of a repository.



# Git Workflow

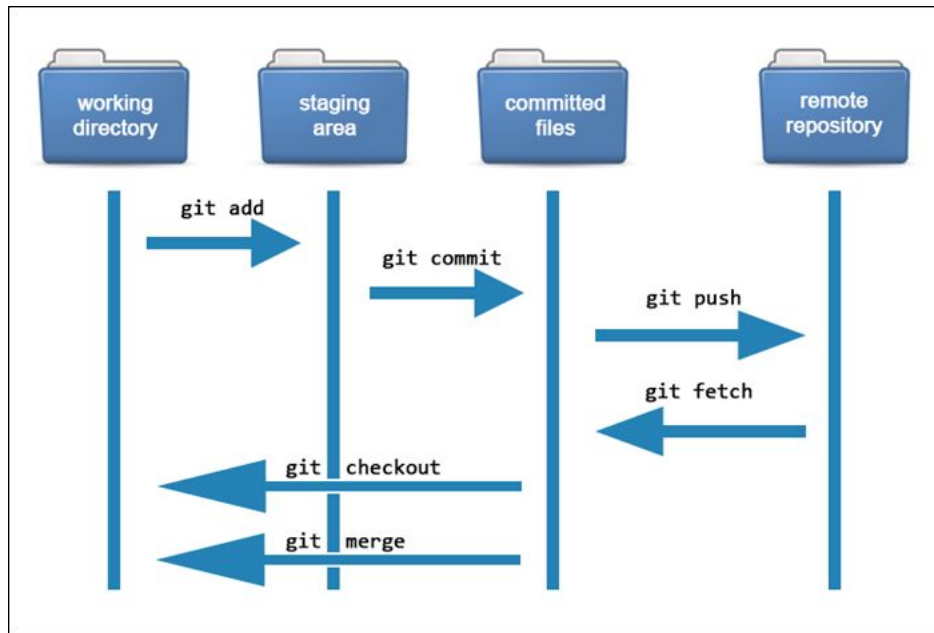
Working Directory → Staging Area → Repository

- Working Directory
  - Where you edit files
- Staging Area
  - Temporary area to group changes for the next commit
- Git Repository (.git folder)
  - Stores the history of your project

# How Git Works

## Git:

- Working Directory
  - Your current files, where you make changes
- Staging Area
  - Git starts keeping track of your files
- Committed Area
  - A screenshot of your working directory, ready to be sent to Github
- Remote Repo
  - Github, Gitlab, Bitbucket, etc



<https://phoenixnap.com/kb/how-git-works#:~:text=Git%20allows%20users%20to%20track,and%20track%20each%20one%20independently.>

# Syntax

## clone

```
git clone <repository url>
```

Retrieves repository from a remote  
location on local machine

## add

```
git add .
```

Add all changes to next commit

## commit

```
git commit -am "message"
```

Creates a snapshot of the changes  
to the code

## push

```
Git push
```

Pushes commit to repository current  
branch

## status

```
git status
```

Shows files ready for next commit

## log

```
git log
```

Shows commit history for the  
branch

# Some Best Practices

- Commit often
  - Debugging
    - It will be easier to identify which change in the code caused an issue, allowing to revert back to a previous version before that change was made
  - Readability
    - Allows for more specific commit messages so that there is a better understanding of what is being changed
- Have good commit messages
  - Be clear and concise when describing the commit
    - Good: "Add test case for functionA"
    - Bad: "added to some functions"

# Some Best Practices

- Use structural elements

- `<type>[optional scope]: <description>`
  - `feat:` allow provided config object to extend other configs
- `fix:` a commit of the type `fix` patches a bug in your codebase
- `feat:` a commit of the type `feat` introduces a new feature in your codebase
- Other types are allowed as well
  - `Build:`, `chore:`, `ci:`, `docs:`, `style:`, `refactor:`, `perf:`, `test:`

- Further Reading

- <https://www.conventionalcommits.org/en/v1.0.0/>

# Branching and Merging

## Branching

- Create a separate “branch” or line of development
- Command: `git branch <branch name>`
- Switch Branches: `git checkout <branch name>`  
or `git switch <branch-name>`

# Branching and Merging

## Merging

- Incorporate changes from one branch to another
- Common commands: `git merge` or `git rebase`
- Use Cases:
  - Experiment with new features
  - Keep your main branch (`main` or `master`) stable

# Pull Requests

## Definition

- A method of submitting contributions to a project
  - Encourages code reviews, discussion, and quality contributions

## Workflow

1. Fork a repository or clone it if you're a collaborator
2. Create a new branch, make changes, commit
3. Push to github
4. Open a Pull Request in GitHub to propose your changes
5. Review and Merge once approved



# Git Exercise

Navigate over to <https://acmatuab.org/assignment>

Follow the instructions to use what you learned

Discover pull requests, branches and merging

Ask for help!

mikegtr@uab.edu

[collincj@uab.edu](mailto:collincj@uab.edu)

## Other Resources

- [Official Git Documentation](#)
- [GitHub Tutorials](#)
- [GitHub Docs](#)

# Internships & Grad School

Getting internships & deciding on Grad School



# Internships

- You can start applying in CS103
  - Big companies typically have freshman/sophomore programs
    - Google STEP, Microsoft Ignite, UberSTAR, Meta University
  - Typically has a low entry barrier
  - Great Experience & Resume boosting
  - Applications start in June/July through September
- Try to TA if you can
  - Shows depth in understanding concepts
- Apply - Network - Go to events

# How to get internships continued pt1

- Applying to freshman/sophomore internships
  - Search for them on Github - "freshman and sophomore internships"
  - <https://github.com/nicolasgarza/early-cs-internships>
- Apply to "regular" internships as well
  - Also available on Github - "Summer <year> internships"
  - <https://github.com/pittcsc/Summer2025-Internships>
- Practice leetcode (or hackerrank)
- Work on **personal** projects - *extremely important*
  - Put them on your Github

# Applying to internships (Resumes)

- Your resume is the most important thing
  - Resumes are the gatekeepers to interviews. To get an interview, you need a good resume
- Fresh/Soph - recruiters cares about format and potential
- Junior/Senior/MS - recruiters care about experience and projects
- Jake's Resume format is the industry standard →

Jake Ryan

123-456-7890 | [jake@su.edu](mailto:jake@su.edu) | [linkedin.com/in/jake](https://www.linkedin.com/in/jake) | [github.com/jake](https://github.com/jake)

## EDUCATION

<b>Southwestern University</b> <i>Bachelor of Arts in Computer Science, Minor in Business</i>	Georgetown, TX Aug. 2018 – May 2021
<b>Blinn College</b> <i>Associate's in Liberal Arts</i>	Bryan, TX Aug. 2014 – May 2018

## EXPERIENCE

<b>Undergraduate Research Assistant</b> <i>Texas A&amp;M University</i>	June 2020 – Present College Station, TX
<ul style="list-style-type: none"><li>• Developed a REST API using FastAPI and PostgreSQL to store data from learning management systems</li><li>• Developed a full-stack web application using Flask, React, PostgreSQL and Docker to analyze GitHub data</li><li>• Explored ways to visualize GitHub collaboration in a classroom setting</li></ul>	
<b>Information Technology Support Specialist</b> <i>Southwestern University</i>	Sep. 2018 – Present Georgetown, TX
<ul style="list-style-type: none"><li>• Communicate with managers to set up campus computers used on campus</li><li>• Assess and troubleshoot computer problems brought by students, faculty and staff</li><li>• Maintain upkeep of computers, classroom equipment, and 200 printers across campus</li></ul>	
<b>Artificial Intelligence Research Assistant</b> <i>Southwestern University</i>	May 2019 – July 2019 Georgetown, TX
<ul style="list-style-type: none"><li>• Explored methods to generate video game dungeons based off of <i>The Legend of Zelda</i></li><li>• Developed a game in Java to test the generated dungeons</li><li>• Contributed 50K+ lines of code to an established codebase via Git</li><li>• Conducted a human subject study to determine which video game dungeon generation technique is enjoyable</li><li>• Wrote an 8-page paper and gave multiple presentations on-campus</li><li>• Presented virtually to the World Conference on Computational Intelligence</li></ul>	

## PROJECTS

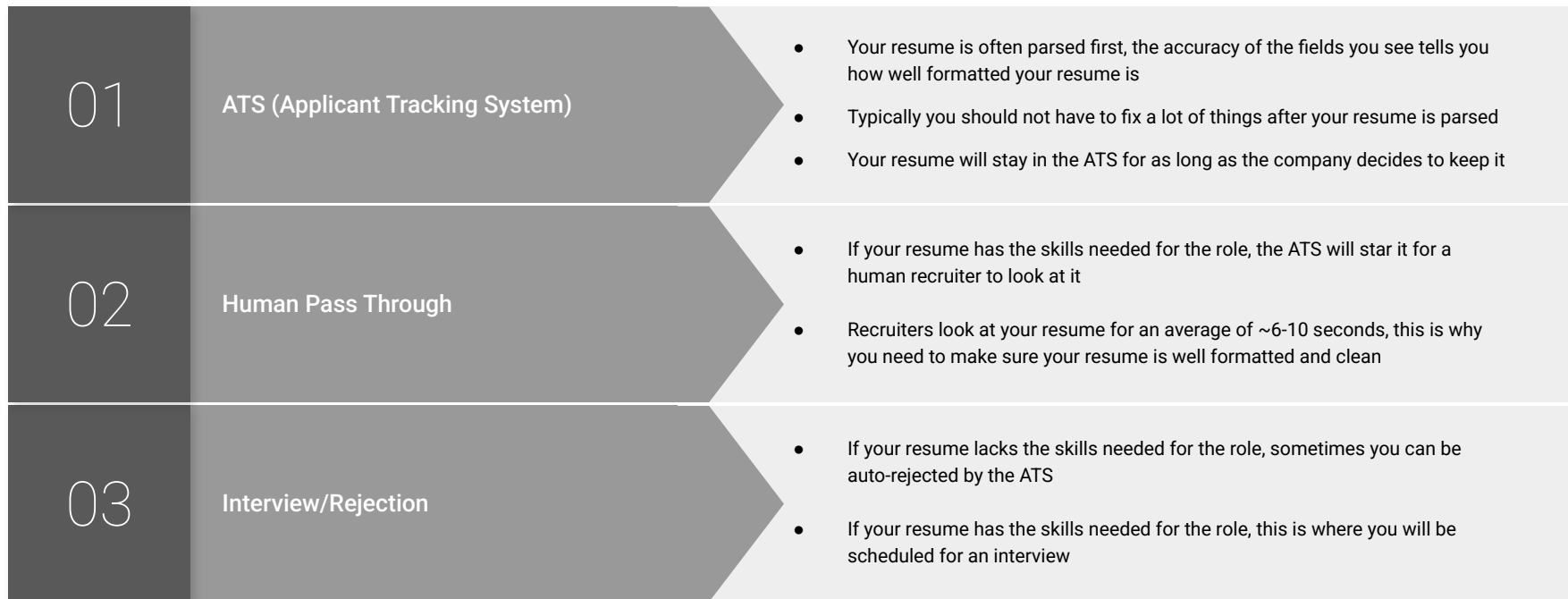
<b>Gitlytics</b>   <i>Python, Flask, React, PostgreSQL, Docker</i>	June 2020 – Present
<ul style="list-style-type: none"><li>• Developed a full-stack web application using with Flask serving a REST API with React as the frontend</li><li>• Implemented GitHub OAuth to get data from user's repositories</li><li>• Visualized GitHub data to show collaboration</li><li>• Used Celery and Redis for asynchronous tasks</li></ul>	
<b>Simple Paintball</b>   <i>Spigot API, Java, Maven, TravisCI, Git</i>	May 2018 – May 2020
<ul style="list-style-type: none"><li>• Developed a Minecraft server plugin to entertain kids during free time for a previous job</li><li>• Published plugin to websites gaining 2K+ downloads and an average 4.5/5-star review</li><li>• Implemented continuous delivery using TravisCI to build the plugin upon new a release</li><li>• Collaborated with Minecraft server administrators to suggest features and get feedback about the plugin</li></ul>	

## TECHNICAL SKILLS

**Languages:** Java, Python, C/C++, SQL (Postgres), JavaScript, HTML/CSS, R  
**Frameworks:** React, Node.js, Flask, JUnit, WordPress, Material-UI, FastAPI  
**Developer Tools:** Git, Docker, TravisCI, Google Cloud Platform, VS Code, Visual Studio, PyCharm, IntelliJ, Eclipse  
**Libraries:** pandas, NumPy, Matplotlib

# More On Resumes

Typically when you upload your resume it follows the following process
















# How to get internships continued

- Network and go to events
  - UAB ACM & UAB WiT regularly hold great events
  - uabacm & uab\_womenintech on Instagram
- Birmingham based events
  - Typically held at [Innovation Depot](#)
  - Tech on Tap, Tech and the City, Sloss Tech, TechTuesday etc - [Calendar](#)
- Get to know your professors/TAs
  - Your TAs will get jobs before you, having a good relationship with them can lead to referrals/advice later



# Internship Interviews

	Startups	Small Cap	Mid Cap	Big Tech	Hedges
Technicals	1-2 (3-4 if VC backed)	1-2	1-2	1-2	~3
Behavioral	1-2	1-2	1-2	1-2	1-2

Companies	 	 	 	   	  
-----------	---	--	--	---	---



# More On Interviews

- Interviews tend to start with an OA (Online Assessment)
  - These are online coding tests that you do on your own time. They are very similar to [Leetcode](#), Zybooks, etc
- One to One Coding Interview: 1-2 interviews
  - Very similar to an OA, except this time you are co-programming with someone at the company. The questions tend to be general technical questions as well as leetcode style questions
  - This [video](#) showcases what a live coding interview is like
- [Unlikely] System Design: 1 interview
  - Although unlikely, some companies have a system design portion. This is designing a program without coding it.
- Behavioral Interview: 1 interview
  - An interview about your experiences, how you work, what you want to get out of the role, etc

# Being successful in your internship

- Keep a weekly record of your wins - share with your manager
  - Internships are short. Keep track of your wins so this allows for you to talk about what you did well every week of your internship.
- Shadow a team member - learn what it takes someone to be successful in the company.
- Talk to everyone on your team
- Do not be afraid to ask questions
  - Your internship will fly by, do not block yourself from progress by being scared to ask questions
- Read - [Marian The Librarian Blog On This](#)

# What about grad school

- Start doing research post-200 levels
  - Start research in your 300 levels
    - Dr. Johnstone is a great person to speak to about UG research
  - UAB Scholarships are available for UG research assistants
- Think about doing a Masters' and/or PhD
  - Research Masters' typically has a good ROI
  - A PhD is what you make it
- Grad schools love to see papers - huge bonus
  - Try to get 1-2 research papers published if you can
  - H-index of  $x > 0$  is great!

# Grad School - Starting Research

- Figure out what you want to do research in
  - Personal projects are a great way to figure this out
  - Generally there are a few research categories of CS
    - AI, Systems, Theory and Interdisciplinary - See [csrankings](#)
- Start reading papers
  - arXiv is a great place to start | [arxiv.org/](#)
- Academia is pretty open
  - Email paper authors with your questions - authors love their research and are always willing to talk about it

# Grad School - Masters' or PhD

- PhD | 4-6 years
  - Professor
  - Research Engineer
  - If you *really really* love research
  - Typically free + stipend
- Masters | 1-2 years
  - Software Engineer - typically 5-15k higher starting pay than Bachelors
  - Research Engineer
  - Lecturer
  - Research Masters can be free
  - Class Based Masters you pay for
- UAB CS has an ABM program
  - ABM allows you to start your Masters before finishing your Undergrad. This allows you to share credits between the two degrees.
  - Email Dr. Zhang or Dr. Johnstone to start enrolling

# Grad School - Applying to Grad School

- Figure out what type of school you'd like to aim for:
  - At top schools, virtually all applicants are qualified
    - Assuming a 5% acceptance rate, if you apply to 10 "top" schools, your probability of going to grad school is  $(1 - 0.95^{10}) = 40.2\%$ .
  - If you apply to ten "top" schools (5% acceptance rate) and ten "regular" schools (10% acceptance rate)
    - your probability of going to grad school is  $(1 - 0.95^{10} \cdot 0.90^{10}) = 79.2\%$ .
- [csrankings.org/](https://csrankings.org/) - great place to find professor and school rankings
- Typically you would like to go to a school for masters with a better CS program than your undergrad program
- Your GPA should be above **3.0** at minimum and above **3.6** for great chance at getting into a good program

# Grad School - Class Based Masters Program

- A class based masters program is a masters consisting of all classes which are all in your program of choice if you do a masters in cs then your masters will only have CS classes
- Advantages:
  - Well-defined course path of 30-35 hours
  - Ideal for people seeking practical, industry-focused skills
  - Can be part-time if seeking a masters while having a job
  - Time to work on personal projects
- Disadvantages:
  - Higher tuition than a bachelors
  - Fewer scholarship options
  - Less exposure to cs research if you want to do a PhD

# Grad School - Research Based Masters Program

- A research based masters program is a masters consisting of 60% classes and 40% research. You spend the first part of your masters doing classes and the end doing research for a thesis.
- Advantages:
  - Offers close mentorship with faculty and research groups
  - Increases potential for publications and conference presentations
  - Serves as a solid springboard to a PhD
  - Builds advanced research and analytical skills - possible research positions in the future
- Disadvantages:
  - Longer and more rigorous than class-based programs.
  - Reduced flexibility for those balancing work or personal responsibilities
  - Limited focus on immediate industry applications
  - Admission can be more competitive



# LinkedIn



# What is LinkedIn

[LinkedIn](#) is a professional networking site, think of it like Instagram but for finding jobs and connecting with people in the professional world. It helps you showcase your work, grow your network and get jobs in the industry

As you embark on your degree, you should make a LinkedIn account to keep up with what is going on in the cs industry as a whole. It can also act as a way to have a public presence for others to find you and see the type of work and degree you are getting.

It's also a great tool for sharing the successes you have during your time in college. From the hackathon wins to great project launches and everything in between. This [blog post](#) does a great job on talking about using LinkedIn as a CS student.

# Plagiarism



# What is considered plagiarism

- Copying work without attribution
  - Always paraphrase
- Copying someone's work who is in (or has taken) the class
  - If you're working together on labs, don't turn in the same exact code
- Using AI tools without adhering to the [Guide for Students on Use of AI tools for UAB CS](#)
- Making class code publicly available without permission

# What happens if you plagiarize

## 1. Course work

The Department enforces a three-strike policy for violations of the UAB Academic Integrity Code by undergraduate and graduate students in their course work:

### 1) First violation:

If it is the first violation by the student in any UAB course, the **minimum penalty** the student receives is a 0 grade for the particular assignment, homework, exam, or project.

### 2) Second violation:

If it is the second violation by the student in any UAB course, the **minimum penalty** the student receives is an F grade for the course in which the second violation occurred.

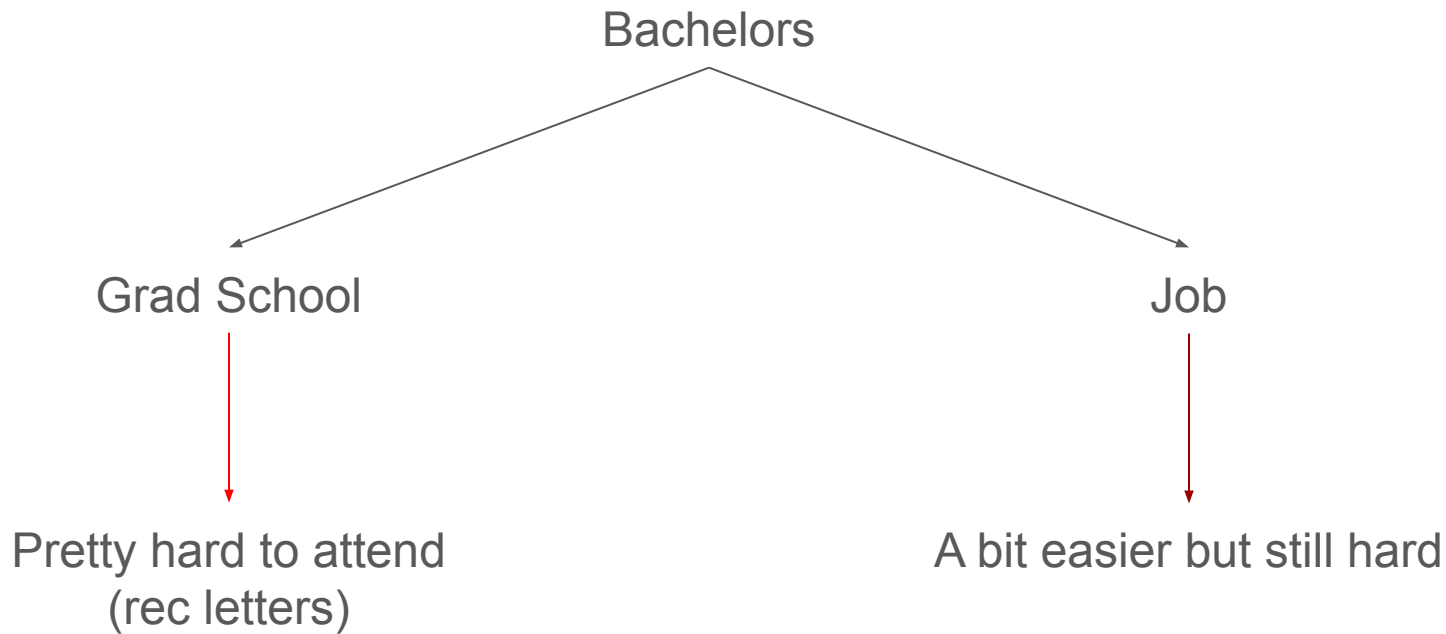
### 3) Third violation:

If it is the third violation by the student in any UAB course, the department will recommend academic probation, suspension or expulsion of the student from the major and from UAB, in addition to an F grade for the course.

# What happens if you plagiarize cont'd

- You can not hold a job at UAB
  - Impacts the jobs you can get as a student/alumni
- You can not ask a professor for a rec/ref letter
  - Impacts your future if you want to go grad school or have a job
- You lose out on points
- Your professors and TAs will be disappointed :(
- You'll have a hard time catching up

# The future



# What happens if you plagiarize cont'd

- You can not hold a job at UAB
  - Impacts the jobs you can get as a student/alumni
- You can not ask a professor for a rec/ref letter
  - Impacts your future if you want to go grad school or have a job
- You lose out on points
- Your professors and TAs will be disappointed :(
- You'll have a hard time catching up



# How to avoid plagiarising

- Start early
  - Read the assignment even if you don't start it
  - Come up with a plan to tackle it
  - Make sure you understand it to make sure you have no last minute questions
- Go to office hours
  - Regularly attending office hours is directly correlated with higher grades
- Ask your TAs questions - Their literal job is to help you
- **Worst case:** a 70 is better than a 0

# AI Tools

How To Use Them To Enhance Learning



# Types of AI Tools

Programmers generally have access to two types of AI tools—general-purpose chatbots like ChatGPT and coding assistants like Copilot. For freshmen and sophomores, relying too heavily on tools like Copilot can hinder learning, as these coding assistants often complete code without explaining the changes they make.

In contrast, ChatGPT offers more flexibility. It can provide code when requested, but it can also explain complex topics without generating code, enabling you to deepen your understanding and enhance their learning experience.

We recommend turning off tools like Github Copilot entirely, especially as a freshman and sophomore student. These next few pages will share some tips on how to use chatbots to help facilitate learning.

# Maximizing Learning

## Ask Specific Questions

- Example: Instead of asking *"What is a loop?"* ask *"Can you explain how a for loop works in Python with examples?"*

## Focus on Conceptual Understanding

- Use prompts like *"Explain recursion step-by-step without providing code."*

## Request Explanations for Code Snippets

- Paste a piece of code and ask, *"Can you explain what this code does in detail?"*

## Learn by Debugging

- Provide an error message and ask, *"What does this error mean, and how can I fix it without providing me code?"*

## Get Action Steps for Assignments

- Provide a piece of your assignment and ask, *"Can you help clarify this for me, do not give code but give me an attack plan"*

## Reinforce Learning with Challenges

- Ask for coding exercises and solutions to practice what you've learned.

# Maximizing Learning Pt2

The most important thing when it comes to AI tools is to make sure you adhere to the assignment and class rules on AI usage

# End

We hope you learned something from this article, if you have any feedback or questions please feel free to reach out to the authors listed on the next slide

# Authors

- Michael Gathara - [links](#)
- Samantha Miller - [linkedin](#)
- Christian Collins - [linkedin](#)
- Rizwan Khan - [linkedin](#)