

CMPS 3390 Homework 1

Spring 2026

Application Development

Application development is the process of creating software programs that perform specific tasks on computers or mobile devices. It involves planning, designing, coding, testing, and maintaining the app to ensure it meets user needs and functions correctly. Developers often work in teams, using programming languages, frameworks, and tools to build apps that solve problems or provide entertainment.

Application

A broad term for any software program designed to perform specific tasks for a user. Applications can run on various devices, including computers, smartphones, and tablets. Examples include word processors, web browsers, and accounting software.

App

A more informal, shortened version of "application." It usually refers to mobile or web applications specifically designed for smartphones, tablets or platform markets. Apps are typically simpler, more user-friendly, and designed for quick, specific tasks, like checking the weather, messaging, or gaming.

Platform

The underlying environment or framework that supports the operation of applications, apps, or services. It provides an entire infrastructure, such as an operating system, a cloud service, or service chain. Platforms enable developers and creators to build, run, and distribute their resources. Platforms often include tools, libraries, and APIs to help developers create resources that are compatible with the platform.

Many projects begin as apps or applications and evolve into platforms.

Main Concepts

Programming Languages & Frameworks

Choosing appropriate languages and frameworks for development.

Examples: JavaScript with React, Python with Django, Swift for iOS, etc

Version Control

Managing and tracking changes in the source code.

Tools: Git, GitHub, GitLab, Bitbucket

Requirement Analysis

Understanding user needs and defining the application's purpose, scope, and features.

Examples: Interviews, use cases, user stories, and requirement specifications

Software Design

Creating the architecture and design of the application.

Examples: System architecture, design patterns, UI/UX design, and data modeling

Collaborative Development

Team Communication, task management, version control, code reviews, documentation.

Tools: Discord, Slack, Github, Trello, Notion, Linear, Markdown

Documentation

Research and development, user feedback, logging, dev history, user manuals.

Tools: Markdown, Sphinx, ReadTheDocs, Notion, Office Suite, README.md

End User Experience

Ensuring the application is user-friendly and intuitive.

Components: Usability testing, responsive design, and accessibility considerations

Graphic Design Basics

Creating a unique look and feel using style guides and custom graphics.

Tools: GIMP, Canva, Inkscape, Krita, Penpot, SVG-Edit, Google Fonts

Client/Server Communications

Connecting the application with external services or other parts of the system.

Examples: https requests, websockets, APIs, integrations, rpc, REST

Data Management

Designing and managing the data layer of the application.

Examples: SQL, NoSQL, local storage, validation, sanitization

Security

Protecting the application from vulnerabilities and ensuring data privacy.

Components: Authentication, authorization, encryption, and secure coding practices

Performance & Testing

Ensuring the application functions correctly and meets requirements.

Components: Unit testing, integration testing, system testing, caching, code optimization

Deployment & Maintenance

Ongoing support, bug fixes, and feature enhancements after deployment.

Components: Monitoring, patch management, and user feedback loops