00 App Design Pacing Guide

Content Area: **TEMPLATE**
Course(s):
Time Period: **Semester**
Length: **20 Weeks**
Status: **Published**

General Overview, Course Description or Course Philosophy

Introduction to App Development is a semester course that introduces students to fundamental concepts of programming and application design through exercises, labs, and app projects. Students will become familiar with the process of debugging and troubleshooting their code; develop new ways of thinking that can be applied outside the classroom. They will learn how to articulate their ideas, collaborate on large complicated projects and how to design with a purpose. Students will have the opportunity to foster their ability to develop and apply computational and design thinking to address real-world problems while design creative solutions. Students will engage not only independently but will work collaboratively to navigate the dynamic digital landscape.

OBJECTIVES, ESSENTIAL QUESTIONS, ENDURING UNDERSTANDINGS

Objectives:

- Explain the importance of data in programming
- Understand how to apply simple mathematical equations in coding
- Explain how swift helps you write safe code
- Determine the outcome of algorithms
- Identify the most appropriate data type for different situation
- Choose the appropriate data structure and algorithm for a specific application
- Solve problems using algorithms and data structures
- Programmers use classes and structures to represent physical objects
- Students should be able to identify common features in multiple segments of code and substitute a single function, data structure, or class
- Name the fundamental data structures (strings, collections, structures, arrays, classes, and dictionaries) and when each one should be used
- Implementing programming solutions using some form of structured design with multiple functions/procedures/modules
- Analyze a large-scale computational problem and identify generalizable patterns that can be applied to a solution.
- Decompose complex problems into manageable sub problems that could potentially be

solved with functions, classes, collections, structures or procedures that already exist.

- Understand the life cycle of an application
- Optimize user experience by using segues and navigation
- Use optionals and guards to create simpler and safer code that checks for nil-value checking
- Transform and idea and sketches into a working application
- Communicate audience and objectives for their application

Essential Questions

- Why is there a need to debug?
- How can computing and the use of computational tools foster creative expression?
- How can computing extend traditional forms of human expression and experience?
- Why is data type safely important?
- Why is a device necessary for real-world testing?
- What relationship does data and programming have?
- How can we store data to solve problems?
- How do computer programs implement algorithms?
- How does abstraction make the development of computer programs possible?
- What benefits and drawbacks to creating structures and classes?
- How does Interface Builder and UIKit group different actions together based on user interaction?
- How does Swift take advantage of flexible variable types?
- How does nested logic alter the execution of code?
- How does designing a system of interacting modules manage the complexity of a program?
- How can users transition between scenes easily?
- How can you check for nil-values?
- What is the relationship between Swift UIKit and Human Interface Guidelines when creating an app?
- How do people develop and test computer programs?
- How do the wants and needs of people influence technology innovation and improvement?
- How do economic, politics, and culture drive the development of new technology?

Enduring Understanding

- Programming uses mathematical and logical concepts.
- People write programs to execute algorithms.
- Critical thinking skills are used to collect and analyze data while debugging.
- Data can not only be stored but manipulated and compared
- Top down programming can be controlled with conditionals
- Programs cannot be written without the use of data
- A variable can only store one piece of data at a time

- An algorithm accomplished a specific task that is made up of a finite set of instructions
- Conditionals determine which part of the code executes based on whether the condition is true or false
- The way statements are sequenced and combined determine the outcome of an algorithm
- Models and simulations use abstraction to generate new understanding and knowledge.
- Algorithms are precise sequences of instructions for processes that can be executed by a computer and are implemented using programming languages.
- Control structures should include conditional statements, loops, and event handlers
- Navigation controllers, tabs, and workflow are used to control how content is organized.
- Swift allows for flexibility in variable creation and assignment.
- Swift has multiple syntax that carries out the same logic.
- Nil-value checking will allow for safer written code.
- Swift allows us to quickly and creatively create minimum viable products for prototyping.
- Well planned and effective branding is critical for a well-designed user experience.
- The design process is an essential process for creating an effective app.
- Programs can be developed for creative expression, to satisfy personal curiosity, to create new knowledge, or to solve problems.
- Programs are developed, maintained, and used by people for different purposes.

CONTENT AREA STANDARDS

CS.9-12.8.1.12.AP.1	Design algorithms to solve computational problems using a combination of original and existing algorithms.
CS.9-12.8.1.12.AP.2	Create generalized computational solutions using collections instead of repeatedly using simple variables.
CS.9-12.8.1.12.AP.3	Select and combine control structures for a specific application based upon performance and readability, and identify trade-offs to justify the choice.
CS.9-12.8.1.12.AP.4	Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue.
CS.9-12.8.1.12.AP.5	Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.
CS.9-12.8.1.12.AP.6	Create artifacts by using procedures within a program, combinations of data and procedures, or independent but interrelated programs.
CS.9-12.8.1.12.AP.7	Collaboratively design and develop programs and artifacts for broad audiences by incorporating feedback from users.
CS.9-12.8.1.12.AP.8	Evaluate and refine computational artifacts to make them more usable and accessible.
CS.9-12.8.1.12.AP.9	Collaboratively document and present design decisions in the development of complex programs.
CS.9-12.8.1.12.CS.2	Model interactions between application software, system software, and hardware.
CS.9-12.8.1.12.CS.4	Develop guidelines that convey systematic troubleshooting strategies that others can use

	to identify and fix errors.
CS.9-12.8.2.12.NT.1	Explain how different groups can contribute to the overall design of a product.
CS.9-12.8.2.12.NT.2	Redesign an existing product to improve form or function.
CS.9-12.8.2.12.ITH.1	Analyze a product to determine the impact that economic, political, social, and/or cultural factors have had on its design, including its design constraints.
	Trade-offs related to implementation, readability, and program performance are considered when selecting and combining control structures.
	Programmers choose data structures to manage program complexity based on functionality, storage, and performance trade-offs.

RELATED STANDARDS (Technology, 21st Century Life & Careers, ELA Companion Standards are Required)

LA.K-12.NJSLSA.L1	Demonstrate command of the conventions of standard English grammar and usage when writing or speaking.
LA.K-12.NJSLSA.L2	Demonstrate command of the conventions of standard English capitalization, punctuation, and spelling when writing.
LA.K-12.NJSLSA.L6	Acquire and use accurately a range of general academic and domain-specific words and phrases sufficient for reading, writing, speaking, and listening at the college and career readiness level; demonstrate independence in gathering vocabulary knowledge when encountering an unknown term important to comprehension or expression.
LA.K-12.NJSLSA.R7	Integrate and evaluate content presented in diverse media and formats, including visually and quantitatively, as well as in words.
LA.K-12.NJSLSA.SL1	Prepare for and participate effectively in a range of conversations and collaborations with diverse partners, building on others' ideas and expressing their own clearly and persuasively.
LA.K-12.NJSLSA.SL2	Integrate and evaluate information presented in diverse media and formats, including visually, quantitatively, and orally.
LA.K-12.NJSLSA.SL3	Evaluate a speaker's point of view, reasoning, and use of evidence and rhetoric.
LA.K-12.NJSLSA.SL4	Present information, findings, and supporting evidence such that listeners can follow the line of reasoning and the organization, development, and style are appropriate to task, purpose, and audience.
LA.K-12.NJSLSA.SL5	Make strategic use of digital media and visual displays of data to express information and enhance understanding of presentations.
MA.K-12.1	Make sense of problems and persevere in solving them.
MA.K-12.2	Reason abstractly and quantitatively.
MA.K-12.3	Construct viable arguments and critique the reasoning of others.
MA.K-12.4	Model with mathematics.
MA.K-12.5	Use appropriate tools strategically.
MA.K-12.6	Attend to precision.
MA.K-12.7	Look for and make use of structure.
MA.K-12.8	Look for and express regularity in repeated reasoning.
VA.K-2.1.5.2.Cr1a	Engage in individual and collaborative exploration of materials and ideas through multiple approaches, from imaginative play to brainstorming, to solve art and design problems.
WRK.K-12.P.4	Demonstrate creativity and innovation.

WRK.K-12.P.5	Utilize critical thinking to make sense of problems and persevere in solving them.
WRK.K-12.P.8	Use technology to enhance productivity increase collaboration and communicate effectively.
WRK.K-12.P.9	Work productively in teams while using cultural/global competence.
TECH.9.4.12.CI.1	Demonstrate the ability to reflect, analyze, and use creative skills and ideas (e.g., 1.1.12prof.CR3a).
TECH.9.4.12.CT.1	Identify problem-solving strategies used in the development of an innovative product or practice (e.g., 1.1.12acc.C1b, 2.2.12.PF.3).
TECH.9.4.12.CT.2	Explain the potential benefits of collaborating to enhance critical thinking and problem solving (e.g., 1.3E.12profCR3.a).
TECH.9.4.12.DC.1	Explain the beneficial and harmful effects that intellectual property laws can have on the creation and sharing of content (e.g., 6.1.12.CivicsPR.16.a).

EVIDENCE OF LEARNING

Formative Assessments

- Feedback/Questioning/Observation
- Student Feedback
- Exit Tickets
- Debuging
- Reflection Questions
- Classwork and/or homework
- Guided practice
- Independent practice
- Exit Ticket
- Check Lists
- Conversations
- Rubrics

Summative Assessments

- Quizzes
- Unit Tests
- Unit App Projects

RESOURCES (Instructional, Supplemental, Intervention Materials)

- Developing in Swift Fundamentals - Teachers Guide

- Developing in Swift Fundamentals - Student Book
- Teacher Resources (slides and answer key to labs and classwork)
- Student Labs and Classwork
- Quick Reference Guide
- Curriculum Outline

INTERDISCIPLINARY CONNECTIONS

Interdisciplinary connections are frequently addressed through modeling existing applications and labs whereby creating solutions and analyzing situations from business, geography, health/fitness, statistics and numerous other fields. Examples can be found in topics specific textbook examples, lab practice, guided projects and digital resources.

ACCOMMODATIONS & MODIFICATIONS FOR SUBGROUPS

See link to Accommodations & Modifications document in course folder.