SAINTS GLOBAL

LEADER GUIDE

PROGRAMMING

INTELLECTUAL CORE

Version 2026.1

PURPOSE & IDENTITY

SKILL BADGE PURPOSE

To develop disciplined problem-solving, safe and ethical computing habits, and practical software-building skill—including agentic programming—through real programs that take input, make decisions, and produce outputs.

DEVELOPMENT CORE: INTELLECTUAL

This badge develops intellectual attributes through focused activities and reflection. Saints will grow in this area while building practical skills.

CORE FOCUSES

- Digital safety and healthy work practices
- Programming history and language evolution
- Inputs, decisions, outputs, and debugging discipline
- Agentic programming with safeguards and human oversight
- Intellectual property, licensing, and honorable software use
- Career awareness and service-minded application

TIME COMMITMENT	RECOMMENDED AGE
4-6 weeks (suggested)	13+

SAFETY CONSIDERATIONS

DIGITAL SAFETY

Follow safe online practices; do not share personal information; use approved, supervised accounts and environments.

ERGONOMICS

Prevent repetitive strain and eyestrain with proper posture, breaks, lighting, and screen habits; stop if pain occurs.

RESPONSIBLE COMPUTING

Programming must not be used to harm others, bypass security, or violate law or policy; agentic systems must include guardrails and human approval.

EMERGENCY CONTACTS

Troopmaster: _____

Emergency: _____

THE DPAR METHOD

Saints Global uses the DPAR method for skill badge completion. As a leader, you should practice DPAR yourself when preparing to teach.

D

DISCOVER

Learn foundational knowledge and concepts. Research, study, and explore the topic.

YOUR ROLE AS LEADER:

- Immerse yourself in the material before teaching
- Study each requirement—understand what AND why
- Anticipate questions saints might ask

P

PLAN

Create a personal action plan with goals and timeline.

YOUR ROLE AS LEADER:

- Design your teaching approach for each requirement
- Gather materials and prepare discussion questions
- Consider how to adapt for different learning styles

A

ACT

Execute through hands-on practice with leader guidance.

YOUR ROLE AS LEADER:

- Shift from teacher to guide—step back
- Create safe space for practice and mistakes
- Model the skills yourself when helpful

R

REFLECT

Review what was learned and share experiences gained.

YOUR ROLE AS LEADER:

- Facilitate meaningful conversations
- Ask open-ended questions, listen more than speak
- Celebrate growth and help saints see their progress

STEP 1: DISCOVER

LEADER PREPARATION

- [ ] Review all DISCOVER requirements thoroughly
- [ ] Gather necessary resources and materials
- [ ] Prepare discussion questions and activities
- [ ] Identify potential challenges saints may face

STEP 1: DISCOVER — TEACHING GUIDE

Requirement 1a: Complete a digital safety briefing approved by your parent/guardian and leader (e.g., a Digital Safety video) and explain two rules you will follow to protect yourself and others online.

HOW TO TEACH

- Have the Saint summarize the briefing in their own words
- Ask for two concrete rules (accounts/passwords, sharing info, safe communication)
- Use a scenario: 'A stranger asks for your info—what do you do?'
- Reinforce that digital safety is part of loving your neighbor

Completion: Saint completes the briefing and states two clear, enforceable safety rules.

Requirement 1b: Discuss prevention and first aid for programming-related injuries (repetitive strain, eyestrain) and demonstrate an ergonomic workstation setup.

HOW TO TEACH

- Have the Saint adjust chair height, screen position, and wrist posture
- Discuss early warning signs and when to stop
- Practice a short stretch routine and eye rest technique
- Emphasize prevention as discipline, not toughness

Completion: Saint demonstrates ergonomic setup and explains prevention and response steps.

Continued on next page...

STEP 1: DISCOVER — TEACHING GUIDE (CONTINUED)

Requirement 1c: Explain the history of programming and language evolution by describing at least three major milestones and why each mattered.

HOW TO TEACH

- Encourage milestones across eras (early computing, high-level languages, internet/open source, mobile/cloud, AI tools)
- Ask: 'What problem did this milestone solve?'
- Tie milestones to accessibility and productivity
- Keep discussion factual and clear

Completion: Saint describes three milestones and explains their significance accurately.

Requirement 1d: List five popular programming languages and describe industries where each is commonly used, then name three programmed devices you rely on daily.

HOW TO TEACH

- Ask the Saint to connect languages to real products (web apps, mobile, embedded, data)
- Avoid ‘this is best’ claims—focus on fit-for-purpose
- Have the Saint point to devices and identify what software controls
- Reinforce that software choices have consequences

Completion: Saint lists five languages with realistic industry uses and identifies three daily programmed devices.

STEP 2: PLAN

LEADER PREPARATION

- [ ] Review all PLAN requirements thoroughly
- [ ] Gather necessary resources and materials
- [ ] Prepare discussion questions and activities
- [ ] Identify potential challenges saints may face

STEP 2: PLAN — TEACHING GUIDE

Requirement 2a: Explain four types of intellectual property (copyright, patent, trademark, trade secret) as they relate to software, and explain licensing vs owning software (including freeware, open source, and commercial terms).

HOW TO TEACH

- Use short examples (code repo, app store, brand logo, proprietary algorithm)
- Ask the Saint to explain what you are allowed to do under each type
- Discuss why honoring terms is integrity
- Have the Saint explain one open-source license concept at a high level (permissions/conditions)

Completion: Saint accurately explains IP types and software licensing/terms.

Requirement 2b: Select three programming languages and development environments you will use for this badge and define a simple input→decision→output project for each.

HOW TO TEACH

- Keep projects small but real—prefer something the Saint can demonstrate live
- Require test cases written as plain language before coding
- Teach debugging as a step-by-step discipline (reproduce, isolate, fix, verify)
- Confirm the Saint can explain the project without reading from notes

Completion: Saint presents three feasible projects with clear input/decision/output and a testing plan.

Continued on next page...

STEP 2: PLAN — TEACHING GUIDE (CONTINUED)

Requirement 2c: Define what 'agentic programming' means and plan an agent workflow that includes tools, constraints, and human approval checkpoints.

HOW TO TEACH

- Use a 'helpful assistant' example: summarize notes, draft a plan, generate code, but require review
- Discuss prompt injection and why agents must verify and constrain
- Have the Saint state a 'stop rule' that triggers human review
- Emphasize that agency increases responsibility

Completion: Saint defines agentic programming and produces a guarded workflow with approvals and failure-mode controls.

STEP 3: ACT

LEADER PREPARATION

- [ ] Review all ACT requirements thoroughly
- [ ] Gather necessary resources and materials
- [ ] Prepare discussion questions and activities
- [ ] Identify potential challenges saints may face

STEP 3: ACT — TEACHING GUIDE

Requirement 3a: Build, debug, and demonstrate Program 1 in Language/Environment #1 that takes user input, makes at least one decision, and produces computed output.

HOW TO TEACH

- Require a live run, not screenshots
- Ask the Saint to predict output before running a test case
- Teach debugging by printing/logging or using a debugger
- Reward clarity and correctness over 'cool features'

Completion: Saint demonstrates correct I/O behavior, explains logic, and shows real debugging discipline.

Requirement 3b: Build, debug, and demonstrate Program 2 in Language/Environment #2 with different input and decision logic than Program 1.

HOW TO TEACH

- Choose a project that highlights the language's strengths
- Require correct handling of invalid input (simple validation)
- Ask the Saint to compare developer experience across environments
- Keep focus on decisions and outputs

Completion: Saint demonstrates the program and explains decisions, tooling differences, and tradeoffs.

Continued on next page...

STEP 3: ACT — TEACHING GUIDE (CONTINUED)

Requirement 3c: Build, debug, and demonstrate Program 3 in Language/Environment #3 with a new type of input/output (e.g., file, JSON, simple UI, or command-line arguments).

HOW TO TEACH

- Encourage a small, useful tool (converter, checklist generator, analyzer)
- Require the Saint to explain data parsing and validation
- Teach 'happy path' vs 'error path' testing
- Keep the scope small but complete

Completion: Saint demonstrates correct alternate I/O, explains flow, and shows verification steps.

Requirement 3d: Implement and demonstrate an agentic program that performs a multi-step task with constraints, tool use, and a human approval checkpoint.

HOW TO TEACH

- Keep the agent in a safe sandbox; no real-world actions without approval
- Require a demonstration where the agent refuses an unsafe or unclear request
- Have the Saint explain the guardrails as part of the demo
- Emphasize reliability: verify, constrain, and review

Completion: Saint demonstrates a working agent with tool use, approvals, logging, and refusal/stop behavior.

STEP 4: REFLECT

LEADER PREPARATION

- [ ] Review all REFLECT requirements thoroughly
- [ ] Gather necessary resources and materials
- [ ] Prepare discussion questions and activities
- [ ] Identify potential challenges saints may face

STEP 4: REFLECT — TEACHING GUIDE

Requirement 4a: Explain what you learned about debugging, discipline, and responsibility when software affects other people, and describe one habit you will keep as you continue programming.

HOW TO TEACH

- Ask for a specific debugging moment that changed understanding
- Discuss how small errors can cause big consequences
- Encourage a single concrete habit (tests, code review, documentation, backups)
- Connect responsibility to honesty and humility

Completion: Saint reflects with concrete examples and identifies a practical habit.

Requirement 4b: Identify three programming-related career pathways and describe the education/training for one you might explore.

HOW TO TEACH

- Include options beyond ‘software engineer’ (QA, cybersecurity, data, embedded, product, IT automation)
- Ask the Saint to match careers to strengths and interests
- Discuss portfolios, internships, and certifications at a high level
- Encourage a next step (talk to a professional, take a class, build a small project)

Completion: Saint identifies three pathways and explains training and a next exploratory step for one.

RESOURCES & CONTACT

RECOMMENDED RESOURCES

- **Saints Global Resource Library** — Online materials and guides
- **DPAR Method Quick Reference** — Printable guide for leaders
- **Child and Youth Program Guidebook** — LDS Church Official Documentation for Children and Youth
- **For the Strength of Youth** — A Guide for Making Choices

SAINTS GLOBAL CONTACT INFORMATION

www.saintsglobal.org

support@saintsglobal.org

Curriculum: curriculum@saintsglobal.org

Thank you for leading Saints Global!

Your dedication makes a difference in the lives of our children and youth.