Quest Of Code

WGU D295 Task1 – Digital Citizenship

# The Quest of Code

## A1. Requirement Brief

**Course Details:** - **Grade Level:** 9–12

**Content Area:** Computer Science / Career and Technical Education (Information Technology Pathway)

**Course Standards:** **National:** CTE (CSTA/IT Pathway)

**College Board Requirements:** AP Computer Science Principles - Create Performance Task

**ISTE Standards** for Students: International Society for Technology in Education (2016)

## A2. Learning Goal

Students will collaboratively design and develop a full-stack web application modeled as a *quest-based learning game* that demonstrates understanding of front-end, back-end, and API integration. Students will apply digital citizenship, computational thinking, and ethical technology practices while creating, deploying, and presenting their prototypes to an authentic community audience.

### A2a. An Elaboration on Goals

In the evolving landscape of computer science education, The Quest of Code transforms programming instruction into an immersive, quest-based experience that merges creativity, collaboration, and technical mastery. Students move beyond routine coding tasks to embark on a narrative-driven project—one that mirrors real-world software development and promotes computational thinking, design, and digital citizenship.

This experience emphasizes both individual mastery and team-based innovation. Students work in collaborative teams to design a functional full-stack application that embodies key software development principles and ethical computing practices. Through this approach, they engage with authentic challenges, applying both technical and creative problem-solving in a quest-ified and meaningful context.

### A2b. Prerequisites

Classroom students are familiar with GitHub Pages, HTML, CSS, JavaScript, and Python. They actively participate in an SDLC (Software Development Life Cycle) that includes code commits, GitHub pushes, integration with team members, and pull requests (PRs) to the instructor’s repository.

### A2c. Background

Prior to engaging in this project, the students were introduced to a project consisting of six modules: activation, demonstration, practice, application, assessment, and integration. They are to create a project that mimics this project’s techniques while learning Full-Stack development (JavaScript interacting with Python/Flask) and utilizing APIs—focusing on login, data storage, retrieval, and Gemini APIs. This learning is modeled as a quest or adventure, with students tracking through the modules as part of an ongoing “Quest of Code” experience.

### A2d. Learning Phases

This new phase begins with *Ideation*, where students conceptualize their projects (shown below), which then moves into *Storyboarding* and *Prototyping*. Each team’s prototype will debut during the school’s **“Night at the Museum”**—an Electives Department showcase attended by families and community members, many of whom are professionals in the technology sector. During this event, students will present their prototypes and gather feedback from the audience.

The general plan includes a simplified *guest login* that allows visitors to traverse through different stages of each project, where interactions capture and log data. This authentic public feedback experience helps student teams test usability and determine whether their prototype is ready to enter the *Development Phase* of the Agile Scrum SDLC.

### A3a. AP Standards Alignment [College Board, 2024]

**AP Computer Science Principles - Create Performance Task:** [Student Task Directions](https://apcentral.collegeboard.org/media/pdf/ap-csp-student-task-directions.pdf)

“Programming is a collaborative and creative process that brings ideas to life through the development of software. In the Create performance task, you will design and implement a program that might solve a problem, enable innovation, explore personal interests, or express creativity.”

### A3b. CTE National Standards [CSTA, 2017]

* **CSTA 3A-AP-13:** Decompose problems and subproblems into parts to facilitate design, implementation, and review of programs
* **CSTA 3A-AP-17:** Systematically design and develop programs for broad audiences by incorporating feedback from users
* **CSTA 3A-AP-22:** Design and develop computational artifacts working in team roles using collaborative tools

### A3c. Digital Citizenship Principles [ISTE, 2016]

* **Empowered Learner:** Students design their learning pathways
* **Digital Citizen:** Ethical use of tools, privacy, and data responsibility
* **Knowledge Constructor:** AI and media literacy integration
* **Computational Thinker:** Efficient and creative problem-solving
* **Innovative Designer:** Iterative design, prototyping, and testing

## A4. Technology Tools

The students will continue to build and grow existing skills; they will begin to develop skills in Digital Literacy and Citizenship related to AI and API usage in coding.

### A4a. Technical Samples

|  |  |
| --- | --- |
| Core Domains | Skill Threads |
| Front-End | HTML, CSS, JS, media presentation |
| Back-End | CRUD operations, database management |
| AI / API Integration | Internet, Deployment, JSON, |
| Data Literacy | Authentic sources, secure data handling, access tokens |
| Communication | Blogging, storytelling, web publishing, and presentation |

### A4b. Learning Activity

#### Group 1: Game Board Progression

Theme: Learning to Code Through Life’s Analogies — ‘Roll your way from traveler to full-stack developer.’

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Lesson** | **Skill or Concept (Exit)** | **Learning Activity** | **Assessment** | **Technology Resource** | **Digital Citizenship Principle** |
| **1. Front-End: “Travel” Module** | HTML/CSS, UI/UX planning | Design a landing page (User View) that guides users on their quest; include buttons, links, and styled interface elements. | Working, responsive interface mock-up | VS Code, GitHub Pages, Chrome DevTools | **Digital Citizen**: Ethical design of user interfaces and accessibility considerations |
| **2. Back-End: “Food / Database Kitchen” Module** | CRUD operations, database schema | Create a small data model to store user progress, awards, or tokens; connect to front-end | Functional CRUD (Create/Read/Update/Delete) implementation | Python Flask, SQLite, Postman API Testing | **Knowledge Constructor**: Secure data handling and privacy protection |
| **3. Integration: “Sports / API Arena” Module** | Fetch, async data handling | Create a module that GETs/POSTs live user data (XP, tokens, progress) | Working API connection and visible data flow | JavaScript Fetch API, JSON, GitHub Actions | **Computational Thinker**: Efficient data processing and API security |
| **4. Reflection & AI: “AI / Code Coach” Module** | AI-assisted debugging, optimization | Use AI tools to identify and resolve code issues or improve features and insure credibility | Reflection report + improved code commit | GitHub Copilot, ChatGPT, AI Code Review Tools | **Empowered Learner**: Responsible AI usage and source verification |

#### Group 2: Data Famine Progression

Theme: Rebuilding Digital Civilization — ‘Your planet has lost its data and knowledge. Restore the civilization through quests of technology and truth.’

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Planet** | **Theme / Analogy** | **Skill Focus** | **Technology Resource** | **Learning Activity** | **Digital Citizenship Principle** |
| AI Planet | Awakening | AI tools for problem-solving and digital literacy | ChatGPT, GitHub Copilot, Gemini API | Build AI-assisted code solutions and evaluate AI responses for accuracy | **Empowered Learner**: Critical evaluation of AI-generated content and responsible AI usage |
| Media Literacy Planet | Awareness | Identifying misinformation, scams, and bias | Python Beautiful Soup, JSON APIs, Fact-checking websites | Create a web scraper to analyze news sources and media libraries to build a truth verification tool | **Digital Citizen**: Information literacy, source verification, and ethical fact-checking practices |
| Secure Data Planet | Rebuilding Infrastructure | Secure database creation and CRUD operations | SQLite, Python Flask, JWT tokens, bcrypt hashing | Design encrypted user authentication system with secure password storage | **Knowledge Constructor**: Data privacy protection and secure coding practices |
| Microblogging Planet | Communication | APIs, posting, and sharing securely | RESTful APIs, JavaScript Fetch, OAuth 2.0 | Build a secure social media platform with user authentication and content moderation | **Computational Thinker**: Secure communication protocols and data transmission |
| Vault Planet | Completion | Synthesis of all skills | Git version control, GitHub Pages, Jekyll deployment | Create final portfolio showcasing all planet achievements with live demonstrations | **Innovative Designer**: Digital portfolio curation and professional presentation |

### A4c. Assessments

This project will have several built-in checkpoints and reviews. T*he Quest of Code represents a transformative approach to computer science education, combining technical skill development with creative problem-solving and ethical technology practices. All these events are recorded on Canvas, LMS.*

* **Checkpoint 1:** Ideation finalization – sync on above
* **Checkpoint 2:** Storyboard review – story sync before code start
* **Checkpoint 3:** Pre-review Prototype – progress in GitHub
* **Checkpoint 4: Event** Participation - feedback collection
* **Summation:** Reflection of feedback with Instructor

### A4d. Technology Resource

Each lesson’s technology resources directly support the main learning goal of “collaboratively designing and developing a full-stack web application modeled as a quest-based learning game that demonstrates understanding of front-end, back-end, and API integration.”

**Learning Goal Component → Technology Resource Alignment:**

* **Front-End Development**: VS Code, GitHub Pages, Chrome DevTools enable students to create and deploy user interfaces that guide users through their quest-based game
* **Back-End Development**: Python Flask, SQLite, Postman API Testing allows students to build server-side logic and databases that store game progress and user data
* **API Integration**: JavaScript Fetch API, JSON, GitHub Actions provide the tools needed to connect front-end and back-end components with live data exchange
* **Collaborative Development**: VS Code, GitHub Copilot, ChatGPT support team-based coding and peer feedback essential for collaborative software development
* **Quest-Based Learning**: All tools work together to enable students to create quest-ified learning experiences that engage authentic community audiences during presentations

### A4e. Digital Citizenship Principles

In build this lesson we have considered many topics in

1. Digital Access is directly correlated to the entire lesson.
2. Digital Communication is the core requirement of the lesson, specifically blogging and building secure chat.
3. All coding projects are published to the Open Coding Society **[OCS, 2025]** GitHub repository which contains an Apache [license](https://github.com/Open-Coding-Society/pages/blob/main/LICENSE).
4. Two AP [College Board, 2024] topics will be covered in these workshops: “5.5 Legal and Ethical Concerns” and “Safe Computing”.

## B. Implementation - E-Learning Environment

### Selected Activity: Front-End “Travel” Module Implementation

**Activity Selected:** Lesson 1 - Front-End “Travel” Module where students design a landing page that guides users on their coding quest.

**E-Learning Implementation Process:**

This activity is implemented entirely in an online, collaborative environment where students work remotely or in hybrid settings. Students begin by accessing the lesson materials through the Open Coding Society web portal, where they find tutorial videos, code examples, and project templates.

**Step-by-Step E-Learning Implementation:**

1. **Initiation**: Students access the lesson through GitHub Pages links sent via Canvas LMS, Online Classroom Calendar, and Slack messaging platform.
2. **Tutorial Phase**: Students read lesson and watch embedded video tutorials on HTML/CSS fundamentals
3. **Practice Phase**: Students use GitHub in conjunction with VS Code on laptop (or cloud-based VS Code) to write and test code, using VSCode live share students can work collaboratively
4. **Collaboration**: Students use GitHub pull requests to share work and receive peer and instructor feedback
5. **Assessment**: Students deploy their landing pages to GitHub Pages for live demonstration
6. **Reflection**: Students document their learning process through GitHub Issues and blog posts

**E-Learning Benefits**: This approach allows students to work at their own pace, access materials 24/7, collaborate asynchronously with teammates, and build a portfolio of work that’s immediately shareable with potential employers or college admissions.

## B1. Digital Tools

### Tool 1: GitHub Projects + Issue with Canvas integrations

**Purpose**: Project management, assignment distribution, and progress tracking

**Implementation**:

* **Setup**: Teacher creates GitHub Projects with milestone-based task boards for each lesson phase
* **Student Access**: Students receive GitHub Issues invitations through Canvas assignment links, automatically creating tracked tasks
* **Development Environment**: Students work on personal laptops using Visual Studio Code with extensions (Live Server, GitLens) or KASM cloud workspaces for Chromebook users
* **Collaboration**: Students use GitHub Issues for peer code review, bug reporting, and feature requests
* **Progress Tracking**: Teachers monitor individual and team progress through GitHub Project boards and Issue completion
* **Assessment**: Teachers review code commits, comment on Issues, and track milestone achievements through integrated dashboards

**E-Learning Advantage**: GitHub integration, real-time progress visibility, authentic project management experience, accessible on any device.

### Tool 2: Local Development Stack + GitHub Pages Deployment

**Purpose**: Full-stack development environment with live deployment pipeline

**Implementation**:

* **Local Setup**: Students install and configure complete development environment on personal laptops:
  + **Visual Studio Code** with essential extensions (Live Server, GitLens, Python, Ruby)
  + **Ruby + Jekyll** for static site generation and local testing
  + **Python + Pip packages** (Flask, SQLite3, Beautiful Soup) for backend development
  + **Git version control** for code management and collaboration
* **Chromebook Alternative**: KASM cloud workspaces provide identical development environment in browser
* **Development Workflow**: Students code locally, test using Jekyll serve, commit changes via Git
* **Deployment Pipeline**: GitHub Pages automatically deploys student repositories to live websites
* **Portfolio Building**: Each project becomes part of student’s professional portfolio with live demonstrations
* **Community Showcase**: Live projects shared during “Night at the Museum” events for authentic audience feedback

**E-Learning Advantage**: Industry-standard development workflow, professional portfolio creation, immediate live deployment, authentic learning environment that mirrors real-world software development.

### C. References

[CSTA, 2017] Computer Science Teachers Association. (2017). CSTA K–12 Computer Science Standards. Retrieved from <https://www.csteachers.org/page/standards>

[ISTE, 2016] International Society for Technology in Education (ISTE). (2016). ISTE Standards for Students. Retrieved from <https://www.iste.org/standards/for-students>

[College Board, 2024] College Board. (2024). AP Computer Science Principles Create Performance Task. Retrieved from <https://apcentral.collegeboard.org/>

**[OCS, 2025]** Open Coding Society. (2025). *Open Coding Society Curriculum Portal.* Retrieved from [https://pages.opencodingsociety.com](https://pages.opencodingsociety.com/)