📋 BRIEF PRD - PyCode AI

Product: Cloud-based Python IDE with AI Assistant
For: TestSprite QA Team
Version: 1.0

---

1. PRODUCT OVERVIEW

What it is:
Web application for writing and executing Python code with AI assistance and data analysis capabilities.

Key Value:

- Write Python code in browser (no setup)
- Run code instantly in sandbox
- AI helps with coding questions
- Upload CSV/Excel → Clean and plot data in single prompt

---

2. TECH STACK

Frontend: React + Monaco Editor + Tailwind CSS
Backend: Node.js + Express
Database: PostgreSQL
Code Execution: Piston API (sandboxed Python environment)
AI: OpenAI/Gemini/Groq API
Hosting: cPanel

---

3. CORE FEATURES

3.1 Landing Page

- Hero section with "Get Started" CTA
- Features showcase
- "Sign In" and "Sign Up" buttons

3.2 Authentication

- **Sign Up:** Email, password, full name → Creates account
- **Login:** Email, password → JWT token stored in localStorage
- **Protected Routes:** Redirect to login if not authenticated

3.3 Dashboard

- Shows all user projects in grid

- "Create New Project" button
- Open/Delete/Rename projects
- Search projects

3.4 Code Editor (Main Feature)

Layout - 4 Panels:

1. **Left - File Explorer (20%)**
    - List of files in project
    - Create/Delete/Rename files
2. **Center - Monaco Editor (50%)**
    - Write Python code
    - Syntax highlighting
    - Auto-save every 30 seconds
    - Multiple file tabs
3. **Bottom - Output Console**
    - Shows code execution results
    - stdout (green), stderr (red)
    - Execution time
4. **Right - AI Chat (30%)**
    - Chat with AI assistant
    - Quick actions: "Explain code", "Fix errors", "Optimize"
    - Send current code as context

3.5 Code Execution

- Click "Run" button → Code sent to Piston API
- Executes in sandbox (10 sec timeout)
- Returns output to console
- Supports pandas, matplotlib, numpy

3.6 Data Analysis (Key Feature)

Workflow:

1. Upload CSV/Excel file
2. AI analyzes structure
3. User prompt: "Clean data and plot sales by region"

4. AI generates complete Python code (cleaning + visualization)
5. User clicks Run → Results shown

Example:

User uploads: sales.csv (date, product, revenue, region)

Prompt: "Clean and visualize revenue by region"

AI generates full code including:

- Data loading

- Cleaning (nulls, duplicates, types)

- Aggregation

- Bar chart visualization

---

4. API ENDPOINTS

Authentication

POST /api/auth/signup - Create account

POST /api/auth/login - Login (returns JWT)

GET /api/auth/me - Get current user

Projects

GET /api/projects - List user projects

POST /api/projects - Create project

GET /api/projects/:id - Get project details

PUT /api/projects/:id - Update project

DELETE /api/projects/:id - Delete project

Files

GET /api/projects/:id/files - Get project files

POST /api/projects/:id/files - Create file

PUT /api/files/:id - Update file content

DELETE /api/files/:id - Delete file

Code Execution

POST /api/code/execute

Body: { code, language, files }

Response: { success, output, error, executionTime }

AI

POST /api/ai/chat

Body: { message, includeCode, codeContext }

Response: { success, message, tokens_used }

Data Upload

POST /api/data/upload

Body: FormData with file

Response: { filename, columns, rows, preview }

---

5. DATABASE SCHEMA

users (id, username, email, password_hash, full_name, created_at)

projects (id, user_id, name, description, created_at, updated_at)

files (id, project_id, filename, content, language, updated_at)

chat_history (id, user_id, project_id, role, message, created_at)

code_executions (id, user_id, code, output, error, execution_time, created_at)

uploaded_files (id, user_id, filename, file_path, columns, rows, uploaded_at)

---

6. USER FLOWS

First Time User

Landing Page → Click "Get Started" → Sign Up → Dashboard →

Create Project → Editor opens → Write code → Click Run → See output

Data Analysis Flow

Dashboard → Open Project → Upload CSV →

Send prompt to AI: "clean and plot" →

AI generates code → Click Run → View cleaned data + chart

AI Assistance Flow

Write code → Get error → Click "Fix errors" in AI panel →

AI analyzes error + code → Suggests fix → Apply fix → Run again

---

7. KEY TESTING SCENARIOS

Authentication

- ✅ Sign up with valid email creates account
- ✅ Login with correct credentials works
- ✅ Invalid credentials show error
- ✅ JWT token stored and used for API calls
- ✅ Unauthorized requests redirect to login

Projects

- ✅ Create project adds to database
- ✅ Projects list displays correctly
- ✅ Delete project removes from DB
- ✅ Open project loads files in editor

Code Execution

- ✅ Simple print() works
- ✅ Pandas operations execute
- ✅ Errors display in red
- ✅ 10 second timeout enforced
- ✅ matplotlib plots generate

Data Analysis

- ✅ CSV upload parses correctly
- ✅ Excel upload works
- ✅ AI generates cleaning code
- ✅ Single prompt creates full analysis
- ✅ Visualization code runs successfully

AI Chat

- ✅ Send message receives response
- ✅ Code context included when checked
- ✅ Quick actions work correctly
- ✅ Code blocks display with highlighting

File Management

- ✅ Create file adds to project

- ✅ Auto-save works every 30 sec
- ✅ Switch between tabs preserves content
- ✅ Delete file removes from DB

---

8. PERFORMANCE REQUIREMENTS

- Page load: < 3 seconds
- Code execution: < 10 seconds
- AI response: < 5 seconds
- Auto-save: No UI blocking
- File upload: < 10MB, < 5 seconds

---

9. SECURITY

- Passwords hashed with bcrypt
- JWT tokens expire after 7 days
- Code runs in sandbox (no network, limited memory)
- SQL injection prevention (parameterized queries)
- File upload validation (type, size)
- Rate limiting on API endpoints

---

10. ENVIRONMENT VARIABLES

DB_HOST=localhost

DB_PORT=5432

DB_NAME=pycode_db

DB_USER=db_user

DB_PASSWORD=***

JWT_SECRET=***

AI_API_KEY=***

PORT=5000

NODE_ENV=production

---

11. SUCCESS CRITERIA FOR TESTING

Critical Paths (Must Work):

1. ✅ User can sign up and login
2. ✅ User can create and open project
3. ✅ Code execution returns correct output
4. ✅ AI chat responds to messages
5. ✅ CSV upload → AI prompt → Complete analysis code generated
6. ✅ File save/load works without data loss

UI/UX:

- Responsive on mobile/tablet/desktop
- No broken layouts
- Loading states show properly
- Error messages clear and helpful

Edge Cases:

- Long code executions timeout properly
- Large files rejected
- Invalid tokens handled
- Network errors show user-friendly messages

END OF PRD

This document contains all functional requirements for QA testing. Focus testing on the 6 critical paths above, then expand to edge cases and performance testing.