KEYVALUE

ScanPay

The US-based payment platform enabling one-tap, on-site payments for field service businesses

Accelerating development with AI: How we delivered a 35-day project in 8.5 days

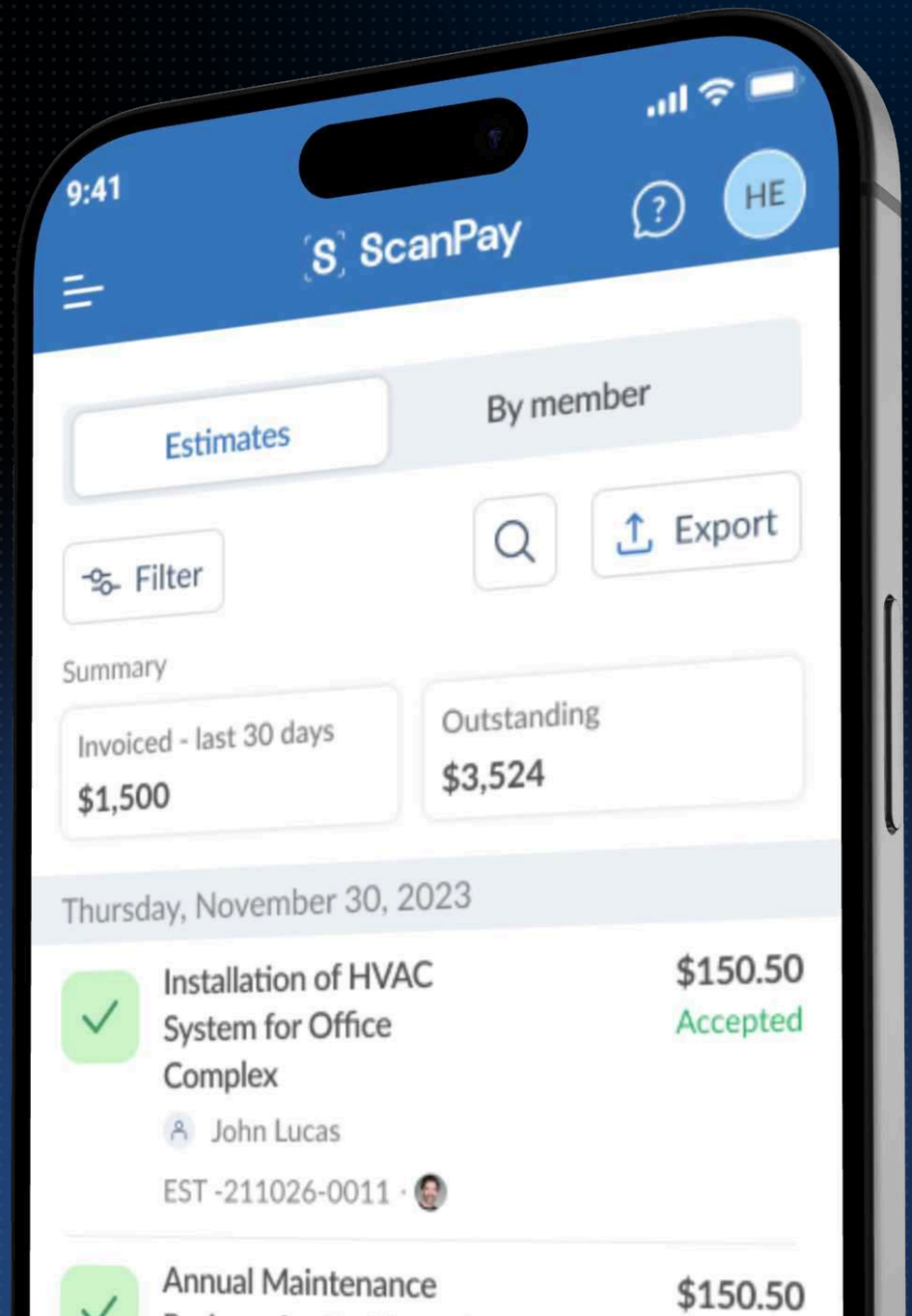
Report type: Case study

Domain: FinTech

Headquarters: United States

Problem statement

Develop an ‘Estimates’ feature to help service professionals estimate their effort and money behind every job. The features had to be developed in the shortest possible time. We had to come up with a solution to build this feature with enhanced efficiency and reduced cost, which usually requires 35 days with 8 engineers.



Challenges

A FinTech app that requires to be robust, scalable, and secure.

2 weeks
fixed deadline

Lean team:
4 engineers

Complexity of
feature replication.

35 days:
historical build time.

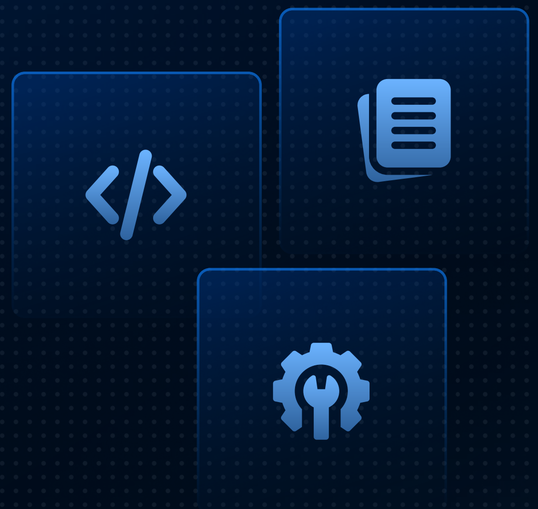
Solution

Phase 1

AI tool selected after evaluation

- **Cursor as primary IDE:** Built on VS Code with advanced code completion and context awareness
- **Claude 3.5 Sonnet:** Used for architecture and planning with its 200k token context window
- **v0:** Implemented for rapid prototyping and boilerplate generation

Solution



Phase 2

Development environment optimization

- **Codebase preparation:** Complete indexing of ScanPay's 200,000 lines of code
- **Tool configuration:** Optimized context window usage with standardized prompting patterns
- **Documentation integration:** Standardized API documentation templates

Solution

Phase 3

Development process

- **Training:** One-day intensive training for team members
- **Planning:** Used AI to analyze requirements, create feature matrices, and generate API contracts
- **Implementation:** Leveraged AI for boilerplate, documentation, and test creation
- **Quality Assurance:** Maintained human code reviews augmented with AI analysis

Results

Project efficiency

Timeline

(76% reduction)

Team size

04 engineers Vs 08 engineers

(50% reduction)

Cost savings

~$60K

Person-days

(88% reduction)

Results

Development velocity

CRUD operations

~70%
faster

API integration

~80%
faster

Frontend components

~60%
faster

UX

25%

Business logic

20%

DevOps

20%

Results

Quality metrics

Test case creation

~85% faster

Test coverage

94% vs target of 90%

Pre-release bugs

07 total vs 25 average

Critical bugs

0 vs 2-3 typical

Factors that helped us succeed

Planning and setup

- Invest time in proper AI tool setup
- Create clear prompting guidelines

Development process

- Maintain regular code reviews
- Document learnings and patterns

Development process

- Define AI interaction guidelines
- Maintain regular sync points