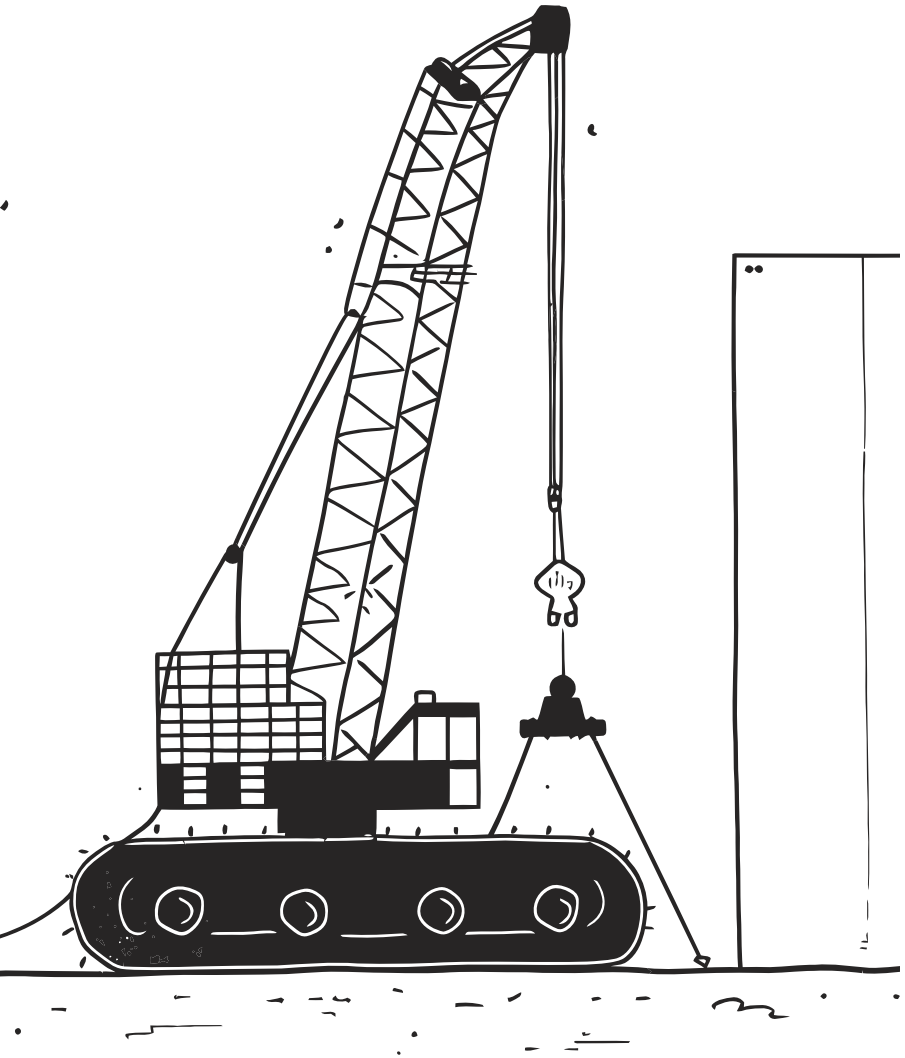
From TDD to Vibe Coding

Transforming development through AI-powered programming

TDD Tutorial: Test-Driven Development



Core TDD Concepts

The fundamental principles that drive test-driven development

What is Test-Driven Development?

- Writing tests before code
- A software development methodology

Problems with Code-First Approach

High defect rates

Poor maintainability

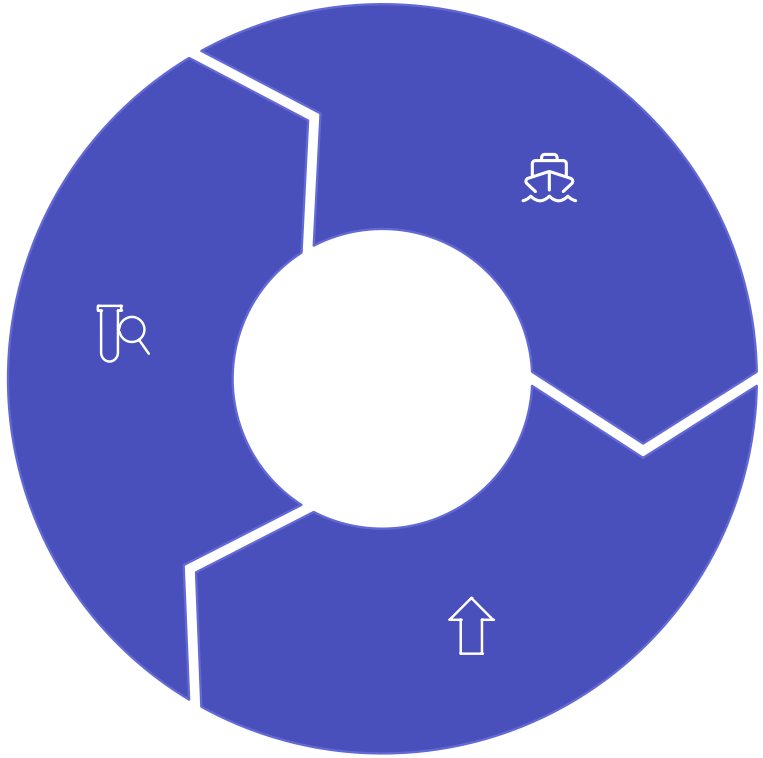
Fear of change

Late testing leads to expensive bug fixes and time pressure

Red-Green-Refactor Cycle

Red

Write a failing test



Green

Write minimal code to pass

Refactor

Improve while tests pass

Benefits of TDD

- Reduces bugs through early detection
- Improves code quality
- Makes refactoring safer
- Ensures code meets requirements
- Builds confidence through feedback

TDD Strategies and Techniques

Approaches to effectively implement test-driven development

Faking It

Start with simple constants instead of complex algorithms

Evolve implementation as understanding grows

Obvious Implementations

Focus on simple implementations first

Allow algorithms to evolve gradually

Don't handle all scenarios upfront

Triangulation

Identify duplication

Extract common methods

Apply DRY principle

Refactor without fear using test safety net

Working in Small Increments

- One small test, one small piece of code
- Just enough code to make current test pass
- Don't anticipate future requirements

Testing Types and Tools

Different testing approaches for comprehensive coverage

Unit Testing

Testing small pieces in isolation

Individual functions or components

Using Jest framework

Integration Testing

Testing multiple components together

Combining units with mocked dependencies

End-to-End (E2E) Testing

- Testing complete user journeys
- Simulating real user interactions
- Using Cypress framework

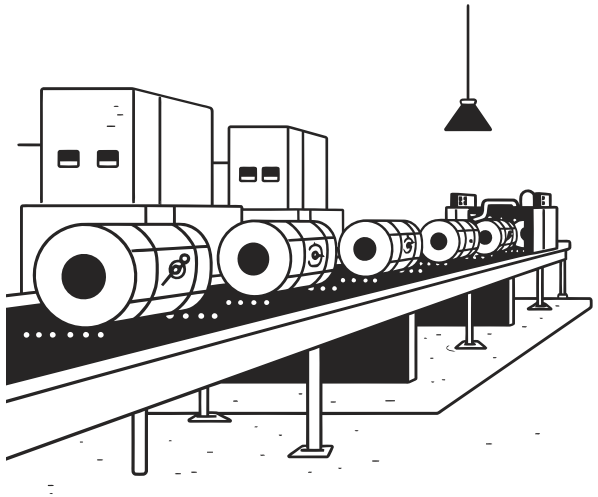
API Mocking

Simulating server responses

Using Mock Service Worker

Fast, reliable, isolated tests

Module Mocking



- Replace real dependencies with controlled alternatives
- Achieve isolation, control, and speed
- Using Jest mocking capabilities

Testing Libraries

Jest

- Testing framework
- describe, it, expect functions

React Testing Library

- Testing React components
- Focus on behavior not implementation

E2E and Mocking Tools

Cypress

E2E testing with visual runner

Mock Service Worker

API mocking for Jest and Cypress

What You'll Learn

Build Real Apps

Create deployed applications using AI assistance

Maximise Tooling

Understand why tooling matters and unlock its potential

AI Communication

Master effective strategies for working with AI

Efficient Building

Develop skills that extend beyond coding

Vibe Coding - Coding with AI

A revolutionary approach that transforms how we think about software development.

Pre-Course Tips for Success

Be Precise

One Task at a Time

Give Coding Agent focused, singular instructions

Break Down Complexity

Divide complex tasks into manageable pieces

Specific Prompts

Make instructions detailed and unambiguous

Stay Organised

01

Keep Projects Tidy

Maintain clean, organised project structure

02

Add Features Step by Step

Build incrementally for better control

03

Test Thoroughly

Verify each addition before proceeding

04

Start Fresh Sessions

Begin new features with clean context

05

Roll Back When Needed

Don't hesitate to revert broken changes

Be Patient

Build Understanding

Learn your app's components gradually

Read Explanations

Study Replit Agent's detailed guidance

Review Before Accepting

Evaluate suggestions thoughtfully

Trust the Process

Allow AI to surprise you with solutions

Remember: Debugging is part of development. Allow time for this essential process.

Understanding Replit Platform

What is Replit?

Replit is a browser-based development environment that requires no installation or setup.

Zero Setup

Runs entirely in your browser with no configuration needed

Built-in AI Tools

AI assistance for every development step

Full-Stack Capabilities

Databases, storage, authentication included

One-Click Deployment

Everything needed for complete applications

Replit's AI Tools

Replit Agent

Automated developer that can:

- Build entire projects from scratch
- Generate project plans
- Make complex code changes
- Work from voice or text input

Replit Assistant

Lightweight tool for:

- Chat and quick edits
- One-off code modifications
- Project questions and guidance

Five Essential Vibe Coding Skills

1. Thinking

Logical Thinking

Understanding basic concepts (like chess rules)

Analytical Thinking

Breaking down problems (playing chess well)

Computational Thinking

Understanding patterns (programming chess rules)

Procedural Thinking

Creating solutions (programming a chess champion)

Focus on computational and procedural thinking to translate complex problems into implementable solutions.

2. Frameworks

Understanding Your Problem Space

1. **Think Through Your App**

   Plan how your application should work

2. **Research Solutions**

   Study existing approaches and common patterns

3. **Ask AI About Frameworks**

   Learn relevant packages and technologies

4. **Accelerate Learning**

   Build understanding whilst developing

> "What frameworks help with drag-and-drop interfaces?"

3. Checkpoints

Version Control Strategy

Automatic Control

Every Replit project has built-in version control

Safe Rollback

Revert failed experiments without losing work

Logical Steps

Break development into manageable phases

Short Sprints

Work in focused, time-boxed iterations

Create Checkpoints

Save progress when features work properly

4. Debugging

Systematic Problem-Solving

01

Understand Your App

Know how your application works

02

Identify Error Location

Pinpoint where problems occur

03

Ask Methodical Questions

Work from first principles

04

Communicate Clearly

Explain issues effectively to AI

Debugging Goals

1. Understand application functionality
2. Locate error sources
3. Determine root causes
4. Communicate issues for resolution

5. Context

Managing AI Context Windows

Context refers to the amount of information an AI can process simultaneously.

Types of Context

Prompts & Instructions

Clear guidance and requirements

Images & Documentation

Visual aids and reference materials

Error Messages

Environment details and diagnostics

User Preferences

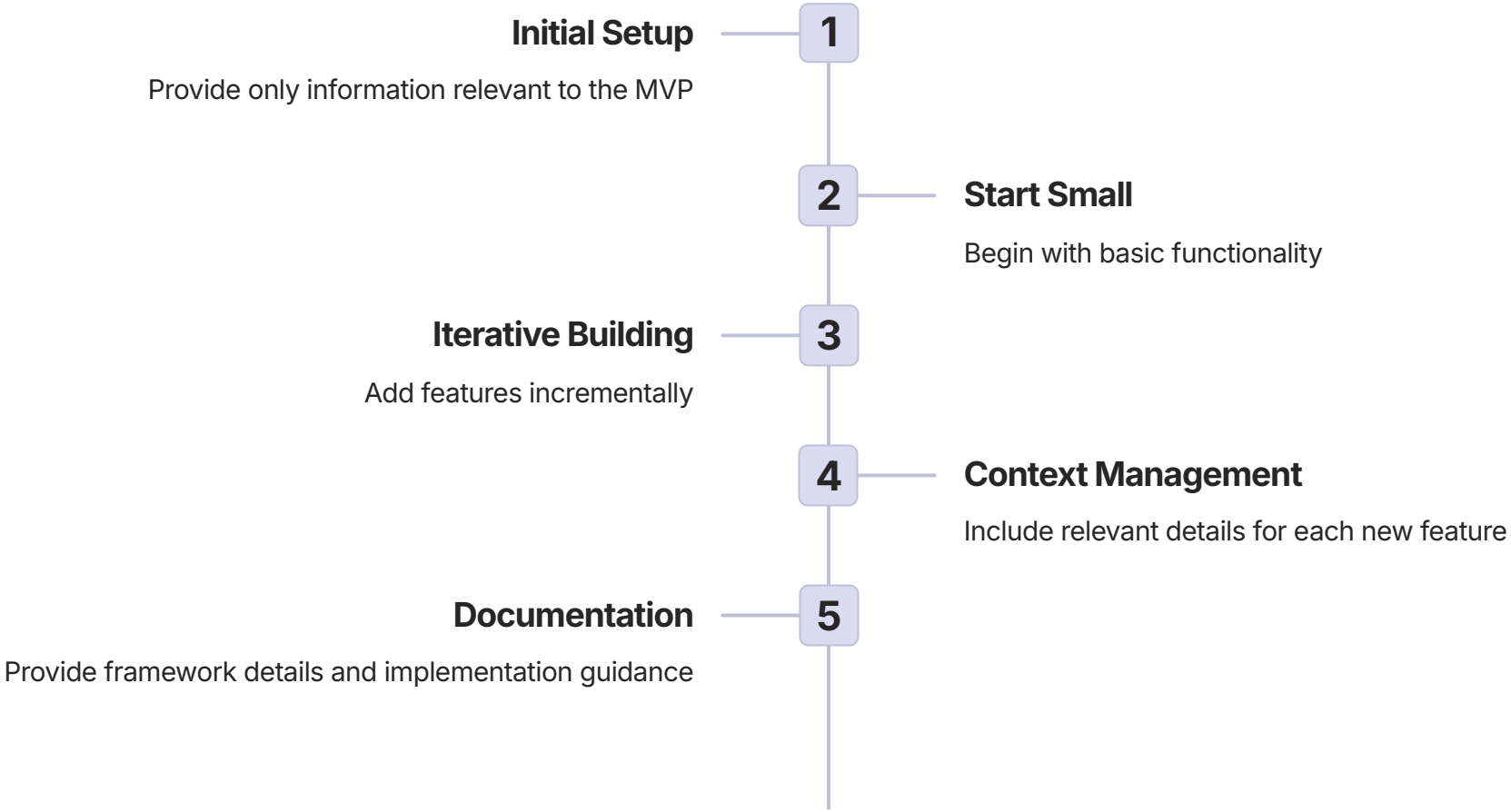
Requirements and specifications

Best Practices

- Keep context relevant to current tasks
- Provide foundational details in initial prompts
- Include specific documentation for novel implementations
- Give explicit implementation details when needed

Development Workflow

MVP Development Process



Feature Development Cycle

Prompt AI → Test → Debug → Checkpoint → Repeat

1. **Prompt**
   Craft clear instructions for MVP or new feature
2. **Test**
   Thoroughly test the implementation
3. **Debug**
   Systematically identify and resolve issues
4. **Checkpoint**
   Save working versions before proceeding
5. **Iterate**
   Move to next feature or enhancement

Error Handling Strategy

Methodical Approach

Approach debugging systematically and thoughtfully

First Principles

Start from fundamental concepts and build understanding

Understand Behaviour

Know your application's intended functionality

Clear Communication

Articulate issues effectively to AI for resolution

Stable Versions

Use checkpoints to maintain working code

Getting Started

Account Setup

01

Visit replit.com

Navigate to the platform

02

Create Account

Use Gmail, email/password, or other options

03

Access Homepage

Use the chat-first interface

04

Explore Sidebar

Manage applications, deployments, settings

First Steps

- Start with homepage chat interface
- Begin with simple, well-defined projects
- Follow development cycle: prompt, test, debug, checkpoint
- Build understanding through AI explanations
- Experiment safely using version control

Replit (https://replit.com/)

Replit

Replit – Build apps and sites with AI

Replit is an AI-powered platform for building professional web apps and websites.