Issue #3: Migrate to TypeScript (or document alternative language choice)

Labels: `enhancement`, `refactoring`, `type-safety`
Milestone: Production Ready
Estimated Effort: 12-16 hours

Description

Migrate the codebase from JavaScript to TypeScript to improve type safety, developer experience, and long-term maintainability. Alternatively, document the decision to use an alternative language (Python, Scala, or Rust) with clear justification.

Why TypeScript?

Current Problems with JavaScript

- ❌ No compile-time type checking
- ❌ Runtime errors that could be caught earlier
- ❌ Limited IDE autocomplete and IntelliSense
- ❌ Difficult to refactor safely
- ❌ No interface/contract enforcement

Benefits of TypeScript

- ✅ Catch errors at compile time
- ✅ Better IDE support and autocomplete
- ✅ Self-documenting code with types
- ✅ Easier refactoring with confidence
- ✅ Better team collaboration
- ✅ Same ecosystem (npm packages)
- ✅ Official Slack SDK has TypeScript support

Technology Decision Matrix

See `TECH_DECISION.md` for full analysis. Summary:

Language	Score	Best For
TypeScript	51/55	⭐⭐⭐⭐⭐ This use case (Slack bot, I/O-bound)
Python	45/55	⭐⭐⭐ Data processing, ML integration
JavaScript	38/55	⭐⭐⭐ Rapid prototyping only
Rust	33/55	⭐⭐ CPU-intensive, systems programming
Scala	31/55	⭐⭐ Complex business logic, enterprise

Recommendation: TypeScript (minimal migration effort, maximum benefit)

Tasks

Phase 1: Setup (2-3 hours)

- [ ] Install TypeScript and type definitions
  `bash`
  ```
    npm install --save-dev typescript @types/node @types/jest
    npm install --save-dev ts-node ts-jest
  ```
- [ ] Create `tsconfig.json` with appropriate settings
- [ ] Update `.gitignore` to exclude `dist/` folder
- [ ] Update npm scripts for TypeScript build
- [ ] Configure Jest for TypeScript testing

Phase 2: Type Definitions (3-4 hours)

- [ ] Define interfaces for data structures
- `LeaderboardEntry`
- `Vote`
- `SlackMessage`
- `CommandResponse`
- [ ] Define types for function signatures
- [ ] Add JSDoc comments where types aren’t obvious

Phase 3: Migration (4-6 hours)

- [ ] Create `src/` directory structure
- [ ] Rename `.js` files to `.ts`
- `index.js` → `src/index.ts`
- `index.test.js` → `src/index.test.ts`
- [ ] Fix type errors incrementally
- [ ] Add proper typing for Slack SDK usage

- [ ] Update imports/exports to use ES modules

Phase 4: Testing & Validation (2-3 hours)

- [ ] Ensure all tests pass with TypeScript
- [ ] Add type-checking to CI/CD pipeline
- [ ] Test compiled JavaScript output
- [ ] Verify bot functionality hasn't changed
- [ ] Update documentation with TypeScript examples

Phase 5: Documentation (1-2 hours)

- [ ] Update README.md with TypeScript setup
- [ ] Update CONTRIBUTING.md with TypeScript guidelines
- [ ] Add type documentation examples
- [ ] Update development workflow docs

Implementation Details

TypeScript Configuration

```json
// tsconfig.json
{
  "compilerOptions": {
    "target": "ES2020",
    "module": "commonjs",
    "lib": ["ES2020"],
    "outDir": "./dist",
    "rootDir": "./src",
    "strict": true,
    "esModuleInterop": true,
    "skipLibCheck": true,
    "forceConsistentCasingInFileNames": true,
    "resolveJsonModule": true,
    "declaration": true,
    "declarationMap": true,
    "sourceMap": true,
    "moduleResolution": "node",
    "types": ["node", "jest"]
  },
  "include": ["src/**/*"],
  "exclude": ["node_modules", "dist", "**/*.test.ts"]
}
```

Type Definitions

```typescript
// src/types.ts

export interface Vote {
  userId: string;
  action: '++' | '--';
}

export interface LeaderboardEntry {
  userId: string;
  score: number;
}

export interface CommandContext {
  command: string;
  ack: () => Promise<void>;
  say: (message: string | object) => Promise<void>;
  client: any; // @slack/web-api WebClient
}

export interface MessageContext {
  message: {
    user: string;
    text: string;
    channel: string;
    ts: string;
  };
  say: (message: string | object) => Promise<void>;
}
```

Updated Package.json

```json
{
  "name": "pp-bot",
  "version": "1.0.0",
  "main": "dist/index.js",
  "scripts": {
    "build": "tsc",
    "start": "node dist/index.js",
    "dev": "ts-node src/index.ts",
    "test": "jest",
    "test:watch": "jest --watch",
    "type-check": "tsc --noEmit",
    "lint": "eslint src/**/*.ts"
  },
  "devDependencies": {
    "@types/jest": "^29.5.0",
    "@types/node": "^20.0.0",
    "@typescript-eslint/eslint-plugin": "^6.0.0",
    "@typescript-eslint/parser": "^6.0.0",
    "eslint": "^8.0.0",
    "jest": "^29.7.0",
    "ts-jest": "^29.1.0",
    "ts-node": "^10.9.0",
    "typescript": "^5.2.0"
  }
}
```

Example Migration: Vote Parsing

```
// Before (JavaScript)
function parseVote(text) {
  const regex = /<@([A-Z0-9]+)>\s*(\+\+|--)/g;
  const matches = [];
  let match;

  while ((match = regex.exec(text)) !== null) {
    matches.push({
      userId: match[1],
      action: match[2],
    });
  }

  return matches;
}

// After (TypeScript)
function parseVote(text: string): Vote[] {
  const regex = /<@([A-Z0-9]+)>\s*(\+\+|--)/g;
  const matches: Vote[] = [];
  let match: RegExpExecArray | null;

  while ((match = regex.exec(text)) !== null) {
    matches.push({
      userId: match[1],
      action: match[2] as '++' | '--',
    });
  }

  return matches;
}
```

Jest Configuration for TypeScript

```
// jest.config.js
module.exports = {
  preset: 'ts-jest',
  testEnvironment: 'node',
  roots: ['<rootDir>/src'],
  testMatch: ['**/__tests__/**/*.ts', '**/?(*.)+(spec|test).ts'],
  transform: {
    '^.+\\.ts$': 'ts-jest',
  },
  collectCoverageFrom: [
    'src/**/*.ts',
    '!src/**/*.test.ts',
    '!src/types.ts',
  ],
  coverageThreshold: {
    global: {
      branches: 70,
      functions: 70,
      lines: 70,
      statements: 70,
    },
  },
};
```

Alternative: Document Language Choice

If choosing Python, Scala, or Rust instead:

1. Create `LANGUAGE_DECISION.md` explaining:
   - Why the alternative was chosen
   - Trade-offs compared to TypeScript
   - Migration plan and timeline
   - Required skill set for team

2. Update `MIGRATION.md` with:
   - Step-by-step migration guide
   - Code equivalents (JS → new language)
   - Testing strategy
   - Deployment changes

3. Update roadmap with revised timeline

Acceptance Criteria

- [ ] All JavaScript code migrated to TypeScript
- [ ] No type errors (`npm run type-check` passes)
- [ ] All tests pass with TypeScript
- [ ] Build produces valid JavaScript in `dist/`
- [ ] CI/CD pipeline includes type checking
- [ ] Documentation updated for TypeScript
- [ ] Code coverage maintained or improved
- [ ] Bot functionality verified unchanged

Testing Checklist

- [ ] Vote parsing works correctly
- [ ] Self-vote prevention still works
- [ ] `/leaderboard` command returns correct results
- [ ] `/score` command works for all users
- [ ] Database integration works (if implemented)
- [ ] Error handling works as expected

Migration Risks & Mitigations

Risk	Impact	Mitigation
Type errors hard to fix	High	Start with `strict: false`, enable gradually
Tests break during migration	Medium	Migrate tests alongside code
Build adds complexity	Low	Document build process clearly
Team unfamiliar with TS	Medium	Provide training, pair programming

Resources

- TypeScript Handbook (https://www.typescriptlang.org/docs/handbook/intro.html)
- TypeScript Deep Dive (https://basarat.gitbook.io/typescript/)
- @slack/bolt TypeScript Examples (https://slack.dev/bolt-js/tutorial/getting-started)
- TypeScript Best Practices (https://www.typescriptlang.org/docs/handbook/declaration-files/do-s-and-don-ts.html)

Dependencies

This issue should be completed before:
- Issue #5: CI/CD workflow (to include type checking)
- Issue #10: Testing improvements (to use TypeScript)