# FUNCTIONAL AND PERFORMANCE TESTING

## 6.1 Performance Testing

1. **Objective**
   * Define the goal of performance testing: to assess responsiveness, stability, and resource usage of the Expense Tracker application.
2. **Testing Environment**
   * Client: Google Chrome (v123+), Vite Dev Server
   * Backend: Node.js (v18+), Express.js
   * Database: MongoDB Atlas Cluster
   * Host System: Intel i5, 8GB RAM, 10 Mbps network
3. **Tools Used**
   * Postman (API testing & benchmarking)
   * Google Chrome DevTools (network analysis)
   * Lighthouse (frontend performance)
   * MongoDB Compass (query efficiency)
   * VS Code debug profiler
4. **Test Cases & Metrics**
   * Login API: response time, token generation speed
   * Income/Expense CRUD: data write/read latency
   * Dashboard endpoint: JSON aggregation and chart loading speed
   * Upload endpoint: Image storage and path generation speed
   * Excel export: File generation and download time
5. **Key Metrics Observed**

| **Feature** | **Avg Time (ms)** | **Max Time (ms)** | **Notes** |
| --- | --- | --- | --- |
| Login API | 220 ms | 310 ms | JWT generation included |
| Add Transaction | 190 ms | 270 ms | Mongo insert latency tested |
| Dashboard Overview | 250 ms | 330 ms | Data aggregation intensive |
| Profile Image Upload | 430 ms | 580 ms | Multer + static path setup |
| Excel Export | 520 ms | 710 ms | XLSX library generation time |

1. **Observations**
   * All endpoints performed within acceptable latency.
   * No memory leaks or 500 errors observed under load.
   * Dashboard aggregation had the highest processing time due to multiple model joins.
2. **Optimization Suggestions**
   * Implement Redis caching for dashboard summaries.
   * Use WebP compression for uploaded images.
   * Optimize MongoDB aggregation pipelines.
3. **Conclusion**
   * The application performs reliably across core features with average latency under 500ms. Frontend assets are optimized through Vite and TailwindCSS. Backend performance is scalable with current architecture and can be improved using caching and indexing strategies.