Dash Duck 软件测试与质量保证报告

1. 测试计划

1.1 测试目标

- 验证应用功能正确性
- 确保用户体验流畅
- 保证数据处理准确性
- 验证多平台兼容性

1.2 测试策略

测试金字塔：

```
    ┌─────────────┐
    │   E2E Tests │  (10%)
    ├─────────────┤
    │Widget Tests │  (20%)
    ├─────────────┤
    │ Unit Tests  │  (70%)
    └─────────────┘
```

2. 测试用例设计

2.1 单元测试用例

```dart
// 位置解析功能测试
group('Location Parser Tests', () {
  test('should parse campus correctly', () {
    expect(parseLocation('南校园教学大楼A101'), equals('南教学A101'));
    expect(parseLocation('珠海校区实验中心B201'), equals('珠海实验B201'));
  });

  test('should handle empty location', () {
    expect(parseLocation(''), equals(''));
  });

  test('should extract classroom number', () {
    expect(parseLocation('北校园C302'), contains('C302'));
  });
});
```

2.2 组件测试用例

```
// 课程表组件测试
testWidgets('Schedule page displays correctly', (WidgetTester tester)
async {
  await tester.pumpWidget(MaterialApp(home: SchedulePage()));

  // 验证页面加载
  expect(find.byType(SchedulePage), findsOneWidget);

  // 验证课程表网格
  expect(find.byType(GridView), findsOneWidget);

  // 验证时间显示
  expect(find.text('08:00'), findsWidgets);
});
```

2.3 集成测试用例

```
// 应用启动流程测试
testWidgets('App launches and navigates correctly', (WidgetTester tester)
async {
  await tester.pumpWidget(DashDuck());

  // 验证首页加载
  expect(find.text('Dash Duck'), findsOneWidget);

  // 切换到成绩页
  await tester.tap(find.text('Grade'));
  await tester.pumpAndSettle();

  expect(find.byType(GradePage), findsOneWidget);
});
```

3. 缺陷跟踪

3.1 缺陷分类

严重程度	描述	处理时间
Critical	应用崩溃、数据丢失	24小时
High	核心功能异常	3天
Medium	次要功能问题	1周
Low	UI细节问题	2周

3.2 质量指标

- **代码覆盖率**: 目标 > 80%
- **缺陷密度**: < 1 bug/KLOC
- **用户满意度**: > 4.5/5.0
- **性能指标**: 启动时间 < 2s

4. 质量保证方法

4.1 预防措施

- 代码审查制度
- 单元测试强制要求
- 持续集成检查
- 静态代码分析

4.2 检测措施

- 自动化测试套件
- 性能监控
- 用户反馈收集
- 线上错误追踪

4.3 改进措施

- 定期测试回顾
- 测试用例更新
- 工具链优化
- 团队技能提升