# QuickEats：项目文档 User3

组别：工具组

# Introduction

1. 引言  
     
    1.1 背景  
     
   在当今快节奏的社会中，餐饮业也在不断进化以适应消费者快速变化的需求。QuickEats，作为一款餐饮及外卖服务平台，其目标是提供高效、便捷的订餐与配送体验。为了实现这一目标，需要一个强大、灵活且易用的软件系统，它能够支持从用户注册、点餐到付款的全过程。只有这样，QuickEats 才能真正意义上提升用户的订餐体验，并帮助餐厅与用户之间建立高效的服务接口。  
     
   这一软件系统的设计始于对用户、餐厅及系统管理者的深入了解。现代消费者的期望不仅限于快速的订餐流程，还需要实时的订单追踪、多样化的支付方式及快速的客户服务。对餐厅来说，高效管理订单、菜品库存与用户反馈也是至关重要的。而管理者更是需要一个完善的后台系统，以支持日常的运营与维护。  
     
   考虑到这些需求，QuickEats 的软件系统应具备以下几个关键特性：  
     
   （1）用户友好性：提供直观、易用的用户界面，让用户在短时间内完成订单流程。  
   （2） 效率：保证高度的互操作性，确保订单从下单到交付的每一个环节都能及时响应。  
   （3）灵活性：支持不同类型的支付方式、多种菜品分类，满足各式用餐场景。  
   （4）透明度：提升用户对订单状态和配送状态的可预见性，通过实时订单追踪提供安心 的用户体验。  
   （5）数据管理能力：建立有效的数据管理机制，以支撑餐馆与管理者对订单、评价的便 捷管理。  
     
   1.2 业务目标  
     
   本文档旨在明确QuickEats这一外卖与订餐平台软件的详细需求，指导软件开发团队构建符合业务场景和用户需求的高质量产品。通过详细阐述用户的实际需求、系统行为及开发者的技术规范，本文档提供一个全面而系统的框架，界定QuickEats平台在User Experience（用户体验）、User Interaction（用户交互）、Backend Operations（后台操作）以及系统整体功能性等多个层面的具体需求。  
     
   推荐阅读者特别关注本文档，它将为开发团队提供明确的项目执行规范和方向，确保QuickEats软件系统开发不仅满足现有需求，还能与时俱进，提供最佳的用户体验。同时，也为未来的扩展留下空间。  
     
    1.3 范围  
     
   本文档覆盖了QuickEats软件系统的所有功能需求，包括但不限于：  
     
   - 用户身份的注册与验证  
   - 订单的生成，支付  
   - 用户购物车管理  
   - 餐厅菜品管理及评价系统  
   - 实时订单追踪、支付历史展示  
   - 管理端的数据处理能力  
     
   同样，本文档也包含了软件系统的非功能需求，如：  
     
   - 用户界面的友好程度  
   - 响应时间、系统的可靠性  
   - 数据安全与隐私保护  
     
   范畴中也包括了对该平台所有功能项的具体约束条件。所有的这些，都确保文档提供完整的信息，为软件开发和产品调试提供必备信息。  
     
    定义、缩写与缩略语  
     
   为保证本规格书一致性，并避免多余的误解，下面是文档中相关术语的简要定义：  
     
   - UC (Use Case)： 用例，表示系统中的一个用户任务或操作。  
   - User： 软件的终端消费者，可在平台上执行各项操作如订餐、付款等。  
   - Restaurant： 提供餐饮服务的商业实体，在QuickEats平台上添加、更新或删除菜品信息。  
   - Dish： 餐厅售卖的每个菜品，在用户订餐时呈现。  
   - Order： 用户在平台上提交的订餐或外卖请求，可能包含一个或多个菜品。  
   - Cart： 购物车是用户的临时存储空间，用于存放准备购买的商品。  
   - Payment Option： 用户在支付时可选择的不同支付方式，如信用卡、支付宝、微信支付等。  
   - Order Record： 记录每次用户下订单的详细信息及状态各改的记录。  
   - Review： 用户对菜品或餐厅服务质量提供的反馈，往往包含评分与文字评论。  
   - Administrator： 平台的管理用户，负责监视并优化平台运营、解决各类问题。  
   - Super Administrator： 平台的最高权限管理者，负责管理员账户的创建与管理、权限设置等重要职责。  
     
    1.4 参考标准  
     
   为了本系统文档的严谨性，以下为本文档所参考的标准及文献资料，并建议读者在阅读时予以参考，以确保本文档相关信息得以更好的理解和使用：  
     
   - ERP系统的设计与实现——这种企业资源计划系统设计主要是针对大型企业的运营管理，为开发类似内部结构的后台支持功能提供了国际参考标准。  
   - RESTful API设计原则——绝大多数组互联网应用使用的API设计风格，该文档可以为需要开发具有通用接口的系统提供指导。  
   - HTML5 & CSS3规范——为平台页面与前端交互设计制定了规范，保证前端用户体验的一致性与最佳实践。  
   - OAuth 2.0 - The Authorization Framework——提供了一套用于WEB应用的对象权限访问认证标准，旨在保证Multi-tenancy环境下的安全性。  
   - W3C Web文档安全性指南——详情指导前端代码实施最佳安全性实践的一套理论准则。  
   - OWASP (Open Web Application Security Project) 安全准则——订立了一系列Web应用程序安全层面的指导原则，内容涉及输入验证、客户管理等。  
   - ISO/IEC 9126软件质量模型——此国际标准定义了软件系统的宏观质量指标，包括功能性、可靠性、可维护等。  
   - ISO/IEC 25010 系统与软件质量需求及评估（SQuaRE）——标准确立扩大了ISO/IEC 9126质量模型，强调更领域化的质量属性，可用于加以规范具体项目执行的质量。

# Overall Description

Product Perspective  
  
QuickEats（以下简称QE）是一款专为用户简化在线餐饮预订体验设计的综合服务平台。愿景是在便利、高效、安全的基础上，让用户能够轻松选择和评价餐厅，并获取高品质的用餐体验。该产品的目标用户群体主要定位为城市中繁忙且注重生活质量的年轻上班族和在校大学生，这些用户不仅希望能够在线预订高品质的餐品，也希望可以通过平台方便地发表评价，与其他人分享自己的用餐体验。市场定位则是一个结合了餐厅管理、订单处理以及用户互动的全方位在线餐饮服务平台。通过融合技术赋能、简化操作流程及优化用户体验等方面，QE旨在构建一个更智能、更友好的餐饮服务生态系统，成为用户不可或缺的在线餐饮解决方案。  
  
 Product Features  
  
QuickEats主要功能模块的设计涵盖了用户、餐厅和管理员三方面的需求。用户端的核心功能有：用户注册和登录、个人资料修改、查看订单历史、评价餐厅或菜品、查看餐厅信息、添加菜品到购物车、从购物车移除菜品、购物车下单、订单取消及支付。在此基础上，用户还可以管理自己的支付方式，并查看支付历史。餐厅端的管理功能包括：添加菜品、更新菜品信息、删除菜品、查看订单、处理订单。管理员端则负责管理订单记录、评价记录，以及管理其他系统管理员。  
  
在系统设计上，各模块间相互联系且紧密配合。用户通过注册、添加菜品到购物车、填写订单、支付等一系列操作，最终形成订单记录。订单确认后，相关信息同步到餐厅管理人员，餐厅及时处理订单，确保用户能按时获取订餐。与此同时，用户可对服务和菜品进行评价，餐厅管理者也能适时更新菜单。管理员通过评价管理体系，不断提升服务质量，保证平台的有效运作。此类模块间的交互，不仅提供无缝的用户体验，还为实现高效的餐厅服务、优质的用户反馈形成了正向循环。  
  
 User Characteristics  
  
QuickEats的目标用户是非洲撒哈拉沙漠以南的年轻人，他们具备一定的智能手机操作能力和日常网络使用习惯，并能够快速适应和掌握在线服务平台的基本功能。这些用户习惯使用手机应用程序来处理日常事务，如购物、预订和支付。大部分目标用户在20至35岁之间，包括城市中的年轻上班族和在校大学生群体。他们生活节奏快，注重效率，往往难以抽出时间亲自去餐馆点餐，因而需要借助类似QuickEats这样的平台以满足快节奏生活。此外，这类用户非常注重个性化体验，偏好视觉效果好、操作便捷并有社交功能的应用程序。因此，在设计QuickEats时，我们将特别考虑用户群体的使用习惯，以最优化用户交互体验为核心原则。  
  
 Competitive Analysis  
  
在众多在线餐饮服务平台激烈竞争的市场环境下，QuickEats能够抓住机会脱颖而出，主要归功于其独特的竞争优势。与一些现有的竞争对手相比，某些APP虽然在订单处理机制上大致相同，但他们在用户和餐厅管理的细节处理上较为乏力。他们通常缺乏对用户个性化需求的重视，例如缺乏充足的用户互动功能及对新餐厅的推广支持，部分APP评分和评价系统设计也过于复杂，增加了用户的操作难度。对此，QuickEats不仅提供了简单直观的菜单浏览体验，还允许用户更直接地高效发表意见或推荐，促进用户的频繁回访与平台忠诚度提高。  
  
- 竞争优势：QuickEats专注于用户行为与需求的改善，在订单支付权限设定及操作环节全面克服了现有的评价系统复杂性问题。此外，我们的平台提供了一套完整的商家管理系统，帮助餐厅实现菜品展示与订单管理，使得商家可以更加灵活便捷地运营，使得服务质量不断优化。  
- 市场痛点：在线餐饮服务市场中仍存在的主要问题包括：支付安全问题、平台可靠性、复杂用户界面设计等。这些问题可能会影响用户体验，降低用户满意度。  
  
 Business Risks and Opportunities  
  
Business Risks  
  
1. 市场竞争加剧：在线餐饮服务平台的快速发展，市场参与者数量迅速攀升，市场竞争加剧。对于QuickEats来说，这可能意味着面临激烈的市场份额争夺战。  
2. 电商平台流量依赖性：平台的用户流量来源等方式可能存在不确定性和波动性，从而影响用户体验和平台稳定性。  
3. 政策法规遵从性风险：需严格遵守关于食品安全、用户信息保护等多方面的法律法规，否则将面临法律风险和潜在的罚款或停业。  
4. 恶意竞争风险：其他竞争对手可能借助商业手段或不当手段对我们进行恶意攻击，这将增加我们的运营成本并影响用户信心。  
   
Business Opportunities  
  
1. 跨平台合作机会：考虑到现有多种媒体类型对用户群体的覆盖，与不同类型的网络媒体进行联合营销能够有助于进一步扩大我们的用户基数，从而提升市场覆盖面和渗透率。  
2. 新兴技术利用机会：借助云计算、大数据和AR等先进技术，有可能为品牌增添更多差异化竞争亮点，拓展业务创新潜力。  
3. 市场扩展机会：除了城市中那些对用餐品质有较高需求的年轻人，我们还看到了潜力市场，比如沿海经济欠发达地区以及其他地区，这些都是市场扩展的机会。  
  
 Design Constraints  
  
在快速发展的大背景下，技术、硬件、法规等因素恰如漩涡中的绳索限制了我们前行，考虑到当前的市场状况，以下几点是开发过程中必须面对和考虑的挑战：  
  
- 技术栈：要求后续加入尽可能做到软硬件开发的高度一体化。未来，QuickEats的重要组成部分均由Web来实现，客户端一部分为原生态App，一部分为WebApp，支持Android、iOS等各大主流设备操作系统。软件架构设计上将采用标准微服务架构，把系统可以详细划分为用户服务、订单服务、菜品服务等，它们之间松耦合，这种设计便于后期开发和演进，降低服务之间的依赖度。  
  
- 硬件平台：针对移动端产品而言，硬件平台限制主要包括屏幕大小、处理器性能、内存容量和存储空间等。我们的应用旨在支持市面上当前主流的手机品牌和硬件配置，保证在这样的条件下表现出色，借此进行不同的适配性调整，保障各品牌Android和iOS设备用户享受良好体验。  
  
- 法规要求：在线餐饮服务平台安全性和隐私保护的重要性不言而喻，我们必须确保能够遵从相关法规标准，例如用户信息保护、合同执行的规范、涉及食品采购审计等，遵守适用的法律与标准，如《个人信息保护法》（GDPR）和关于数据隐私的本地法律。  
  
 Assumptions and Dependencies  
  
在网站及应用程序的开发过程中，通常会对一些设定或预期作出假设和依赖：  
  
- 主要依赖项：假设移动设备具备访问互联网的能力，并能够通过网络数据为本次提供服务。另外，还依赖于使用成熟的前端和后端框架，以及高效的第三方库或服务，这些都将对开发周期、成本、乃至最终用户体验产生不可忽视的影响。本项目依附于数据库服务如MySQL，协同办公软件等第三方软件服务，助力开发的顺利进行。  
  
- 技术框架限制：假设技术文档编写将基于成熟的Java SpringBoot框架，这是一个被广泛接受的编程语言和编码技术，能够显著降低开发难度和复杂度。同时，预期开放的源代码库能够带来开源技术的支持，比如图像处理技术、前端JavaScript框架等，但需注意实现时目标与代码框架之间可能出现兼容性问题，需要开发者及时作出调整对策。  
  
依赖关系：假设在后期维护过程中，本软件将严格依赖于配套的数据监控、性能管理以及服务宕机的预警系统，并通过多种协议进行通信，来确保在线餐饮服务的流畅性、稳定性和安全性。此部分的依赖关系，包括软件与硬件配置不同系统之间的协调合作以及其他第三方服务的有效联动等，对平台性能检修/监视起到重要支撑的作用。

# Functional Requirement

基于提供的系统描述、ER图、用例描述规则，以下是QuickEats餐饮及外卖应用功能需求的详细描述：  
  
- 用户注册  
 描述：新用户可以在QuickEats创建账号。  
 输入：姓名、手机号码、电子邮箱以及密码等基本个人信息。  
 输出：用户账户存储于系统数据库中，用户接收确认邮件或短信。  
   
- 用户登录  
 描述：已经注册的用户使用注册信息登录。  
 输入：电子邮箱和密码。  
 输出：用户成功登录并跳转至其个人主页。  
  
- 用户修改个人信息  
 描述：成功登录的用户可以修改自己的个人信息。  
 输入：用户的登录状态，新的或修改后的个人信息。  
 输出：更新后的用户个人信息更新并存储到系统数据库中。  
  
- 用户查看订单历史  
 描述：用户可以查看之前的订单记录。  
 输入：用户的登录状态。  
 输出：包含用户订单记录的列表及订单详情视图。  
  
- 用户评价餐厅或菜品  
 描述：用户可以在系统中对已经完成订单的餐厅或菜品进行评价。  
 输入：用户的登录状态，选取需要评价的订单、填写评分及评论。  
 输出：在数据库中存储用户的评价推送反馈。  
  
- 用户查看餐厅信息  
 描述：用户可以浏览各餐厅的详细信息。  
 输入：用户浏览动作。  
 输出：展示餐厅的详细信息，包括地址、评分、营业时间等。  
  
- 餐厅添加菜品  
 描述：餐厅管理员可添加新的菜品以供浏览和订购。  
 输入：管理员登陆状态，新的菜品的相关信息（如名称、价格、图片、描述）。  
 输出：新的菜品信息存储到系统数据库中，并显示在餐厅对应的菜品列表。  
  
- 餐厅更新菜品信息  
 描述：餐厅管理员可修改已有菜品的信息。  
 输入：管理员登陆状态，需要修改的菜品信息。  
 输出：更新的菜品信息存储到数据库中，更新菜品详情和菜单显示。  
  
- 餐厅删除菜品  
 描述：管理员可以移除不需要的菜品。  
 输入：管理员登录状态，选择需要移除的菜品。  
 输出：与该菜品相关的信息从数据库中删除。  
  
- 餐厅查看订单  
 描述：餐厅管理员可以查看所有关联该餐厅的订单及细节。  
 输入：管理员登录状态。  
 输出：展示所有关联该餐厅的订单列表以及订单详情。  
  
- 餐厅处理订单  
 描述：管理员可以确认或拒绝订单。  
 输入：管理员登录状态，选择订单的接受或拒绝。  
 输出：更新数据库中订单的状态，并向用户推送处理结果通知。  
  
- 菜品添加到购物车  
 描述：用户可将菜品添加到购物车以备结算。  
 输入：用户登录状态，选择菜品的动作。  
 输出：将菜品信息存储到购物车相关的数据库中。  
  
- 菜品从购物车移除  
 描述：用户可以从购物车中移除某个菜品。  
 输入：用户登录状态，选择移除的菜品。  
 输出：对应的数据从购物车记录中删除。  
  
- 购物车下单  
 描述：用户确认购物车中的物品并生成订单。  
 输入：用户选择的配送地址及支付方式，购物车状态。  
 输出：创建新的订单记录，触发支付流程。  
  
- 订单取消  
 描述：用户可能想取消未支付订单。  
 输入：用户登录状态和要取消的订单。  
 输出：订单状态更新为已取消，购物车状态恢复。  
  
- 订单支付  
 描述：用户支付已生成的订单。  
 输入：用户登录状态，选择的支付方式，订单详情。  
 输出：支付处理后的状态和订单状态更新为已支付，更新支付记录。  
  
- 支付方式设置  
 描述：用户可以设置默认的支付方式。  
 输入：用户登录状态，支付相关的详细信息。  
 输出：支付方式信息存储到数据库中，并设置为默认。  
  
- 查看支付历史  
 描述：用户可以查看并了解其支付历史。  
 输入：用户登录状态。  
 输出：用户的支付历史信息列表，包括每笔支付的详情。  
  
- 发布评价  
 描述：用户可以发布餐厅或菜品的评价。  
 输入：用户登录状态及待评价的订单详情。  
 输出：评价发布，并存储到数据库中，更新相关店铺或菜品的评分。  
  
- 管理评价  
 描述：特殊的后台管理员可以查看并管理所有评价，包括潜在违规或恶意评价的删除。  
 输入：管理员登录状态，需要管理的评价信息。  
 输出：更新评价列表，包括评分和可见性。  
  
- 查看订单记录  
 描述：系统管理员可以查看所有地方的订单记录。  
 输入：管理员登录状态。  
 输出：显示所有订单记录及其详情。  
  
- 添加管理员  
 描述：超级管理员可以新增管理用户的账户。  
 输入：超级管理员的登录状态，新管理员信息。  
 输出：管理员账户信息存储到数据库中，包含权限设置。  
  
- 管理管理员  
 描述：超级管理员可以编辑已有管理员的账户信息或删除不活跃管理员账户。  
 输入：超级管理员登录状态，需要编辑或删除的管理员账号及其新信息。  
 输出：信息更新或管理员账号状态删除对应的数据库和系统状态更新。  
  
这些功能是基于给定的用例，并吻合数据模型来设计的，确保它们能够相互协作，共同构建一个能够满足描述中功能需求的高效系统。

# External Description

4. External Interface  
  
外部接口定义了QuickEats软件系统与外部环境（如硬件、其他软件子系统、数据库、第三方服务等）之间的交互需求。为了确保系统的各个部分能够流畅且高效地协同工作，本节详细描述了接口的类型、通信协议、数据格式等细节。  
  
 4.1 用户界面 (UI) 接口  
  
- 类型：图形用户界面 (GUI)  
- 描述：提供友好的客户端用户界面 (Web 和 Native App)，确保用户能够简易地浏览菜单、添加商品、执行支付等操作。  
- 输入/输出：用户界面接收用户的交互指令，输出包含菜单、商品详情、订单状态等信息的页面。支持触控、点击等交互方式。  
- 数据格式：数据呈现形式包含文本、图像、音频等多形式媒体格式。交互数据格式使用JSON，响应数据格式同样为JSON。  
  
 4.2 外接支付系统接口  
  
- 类型：API接口  
- 描述：与外部支付处理系统进行交互，支持多种支付方式如银行卡、支付宝、微信支付等。  
- 通信协议：RESTful API设计，确保请求和响应遵循HTTP/HTTPS协议。  
- 数据格式：使用JSON格式进行数据交换，确保跨不同支付平台项目的一致性和可解析性。  
  
 4.3 数据库接口  
  
- 类型：ORM接口  
- 描述：系统运用Java SpringBoot框架对人体与关系型数据库（如MySQL）之间的交互进行管理。  
- 通信协议：JDBC和其他相关数据库都有SQL语言操作。  
- 数据格式：数据库层的数据存储和传输采用SQL结构化查询语言。具体实施中各实体的数据格式根据模型设计以实体类的形式进行多态封装。  
  
 4.4 第三方API接口  
  
- 类型：RESTful API接口  
- 描述：与其他第三方服务对接（如地图服务、天气服务等），以提供附加服务。  
- 通信协议：支持HTTP/HTTPS通信协议。  
- 数据格式：基于数据传输协议 (如 JSON) 结构化数据来接收和响应各种API调用。  
  
 4.5 系统监控接口  
  
- 类型：API接口  
- 描述：为了实时监控系统性能，对接监控系统（如Prometheus, Grafana等）。  
- 通信协议：遵循HTTP/HTTPS协议或根据监控系统的具体请求协议。  
- 数据格式：提供格式化的监控数据 (如JSON, Protocol Buffers)，用于可视化实时系统的运行状态和性能数据。  
  
 4.6 开放接口 (API)  
  
- 类型：RESTful API接口  
- 描述：为了开发者和第三方服务能够扩展平台，提供了开放接口。  
- 通信协议：遵循HTTP/HTTPS协议和REST原则。  
- 数据格式：支持JSON、XML等格式。  
- 认证：针对API调用的安全管理使用OAuth 2.0协议。  
  
 4.7 社交媒体接口  
  
- 类型：API接口  
- 描述：存允许用户分享其评价或对菜品的喜好到Facebook、微博、微信等社交媒体。  
- 交流协议：第三方社交媒体API的特定通信协议。  
- 数据格式：具体根据第三方平台的需求确定格式。  
  
 4.8 邮件接口  
  
- 类型：API接口  
- 描述：通过使用第三方邮件服务接口，QuickEats可以发送验证邮件、订单确认邮件、余额提醒等。  
- 通信协议：SMTP、HTTP/HTTPS协议。  
- 数据格式：邮件内容基于HTML格式构建，部分纯粹文本内容为纯文本。  
  
通过上述外部接口的设计，QuickEats系统能够高效接入和利用各种外部服务和资源，确保系统具有良好的扩展性和灵活性，能够在未来的发展中持续满足用户不断变化的需求。

# Non-Functional Requirement

第五章：非功能需求  
  
 5.1 性能要求  
  
 5.1.1 响应时间  
QuickEats需要确保前端操作的即时响应性，具体指标如下：  
  
- 用户界面响应时间：核心操作（如用户注册、支付、浏览菜单等）的响应时间应不超过3秒。页面加载时间不超过5秒，以确保用户可以快速完成相应的操作，从而获得顺畅的使用体验。  
- 数据处理与查询响应时间：系统在处理后台订单状态更新及信息查询等事务时，需保证数据响应时间不超过7秒。特别是在高峰时段，订单处理时间不超过10秒，以确保快速异步消息传递和前端页面的即时更新，从而维持数据的一致性和反馈的及时性。  
- 数据写入持久化时间：包括注册信息、付款信息和其他用户生成内容等，写入持久化存储的时间不超过5秒，以确保用户数据的安全性和系统的稳定性。  
  
 5.1.2 吞吐量  
系统需确保在适用负载条件下能够处理大量请求，并避免性能瓶颈。  
  
- 订单处理吞吐量：每分钟能处理至少1500个有效订单，保证用户提交订单后系统能尽快响应和确认，从而减少订单积压。  
- 用户登录吞吐量：系统在正常负载下每秒支持至少200次有效登录尝试。  
- 用户注册吞吐量：系统在正常负载下每秒支持至少100次有效登录尝试。  
  
 5.1.3 并发用户数  
在高负载情况下（如周末、法定假日等）系统能够处理以下数量级的并发访问：  
  
- 并发访问：系统应能够处理至少3000个并发用户访问、查看菜单、操作购物车等，以确保在高峰期用户体验的流畅度不受影响。  
  
 5.2 可用性  
为确保用户可以随时随地进行交易和信息访问，QuickEats需具备以下高可用性指标：  
  
- 系统可用性：Daily operational availability: 99.95%  
 系统需达到极高的每日平均在线可用率，确保只有极少的不可预见性宕机事件。  
- 访问连续性：Sequential operational clients functioning reliability: 99.9%  
 用户持续使用过程中，系统应达到连续工作可靠性高于99.9%，防止系统中断带来的负面用户体验。  
- 失效恢复能力：Quick recovery time  
 系统在发生故障后，恢复到正常可用状态所需时间不得超过2分钟。恢复顺序应首先修复商业交易处理，其次为信息资源访问，以优化用户体验。  
  
 5.3 可靠性  
可靠性是确保QuickEats在所有预期操作条件下能够正常工作不可或缺的：  
  
- 用户交易可靠性：用户交易（如支付、订单提交等）的成功率需达到99.9%，并在安全情况下保护用户数据。  
- 数据持久性：仅在特殊情况下（如用户主动删除、平台需要紧急维护等）允许数据丢失。数据丢失率需控制在0.001%以内。  
- 系统恢复可靠性（间隔24小时）：当遭遇系统重启或升级时，数据和功能的恢复可靠性需达到99%，确保重启后恢复时长不超过30分钟。  
  
 5.4 安全性  
QuickEats的系统设计必须符合最高的数据安全和隐私保护标准，以确保用户信息和交易的安全。  
  
- 数据传输加密：所有敏感数据通信（如支付信息、用户个人信息等）应采用TLS 1.2以上的技术进行加密。  
- 数据存储加密：用户敏感数据（如密码、支付信息等）和业务关键数据应使用AES 256位密钥进行加密存储。  
  
  
  
 5.5 可维护性  
面对各种技术或者业务需求快速响应与实施，QuickEats应具备高效高效的可维护性。  
  
- 模块化设计： 系统应采用微服务架构，各模块之间的关联尽可能平滑，具有足够的解耦水平，可以在不影响其他模块的情况下进行集中化维护。  
- 适应冗余和过载的能力：系统应能够自动检测吞吐量的激增，从而活性根据需要分配资源。  
- 软件文档：提供详细的组件和模块级别的文档与开发指南，指导开发人员和其它利益相关者理解系统并执行扩展性加固。  
  
 5.6 可移植性  
考虑到跨平台应用对提升用户覆盖及支持业务扩展的基础条件，QuickEats需具备：  
  
- 适应性： 快速迁移到不同云平台的策略,叠加多种移动设备可以访问该系统，如从iOS系统迁移到Android系统，或迁移至不同的云服务供应商。  
- 兼容性： 系统设计力求简化，以适应未来设备的发展趋势。  
- 安装性与本地化： 系统应具备可视化的安装程序，以支持轻松安装于不同操作系统，并能依据不同区域环境进行本地化配置。  
  
本软件需求规约说明书中的非功能性需求部分规定了QuickEats软件系统在性能、可用性、可靠性、安全性、可维护性及可移植性方面的关键要求。这些指标的设计旨在确保系统不仅在技术层面上达到高标准，而且在用户体验和服务质量上能够实现最优，以满足现代消费者对餐饮平台的复杂需求。同时，这些标准支撑系统能在未来不断变化的技术环境和技术需求下保持领先和可行性。这些内容旨在为开发团队勾画制定性能目标及进行系统验证测试的准则，力图打造出一个高效、安全、易于使用的快速餐饮服务平台。