Design Document

**ChequeMeOut**

notsirkApps

**Prepared By**

|  |  |
| --- | --- |
| DOCUMENT OWNER(S) | Organization role |
| Kriston Sanders | Project Manager |

**Version Control**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author(s) | Change description |
| 1.0 | 5/28/20 | Kriston Sanders | Document created |
| 1.1 | 6/5/20 | Kriston Sanders | Generally revisions; stubbed testing section |

**Approvals**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Approving party | Signature |
| 1.1 | 6/5/20 | Kriston Sanders | Kriston Sanders |

| 1. Introduction |
| --- |
| * 1. **Purpose** * Reduce wait times/close contact, and eliminate the need to split checks * Launch product that is fun to use and intuitive * Create product to highlight skills and impress potential employers * Be transparent about data collection/use * Reduce wait times/close contact, and eliminate the need to split checks * Accommodate non-users * Low adoption cost   1. **Document Definitions** * CMO- ChequeMeOut abbreviation * Diner- customers of a restaurant or bar * User- a diner who utilizes the mobile application * Non-user- a diner who does not use the mobile application * Open diner pool- a subgroup in the mobile app to aggregate orders for non-users * Server- employee who interacts with the diners and tracks orders * Kitchen- employees who prepare food * Owner- the owner of the bar or restaurant * Diner app- the mobile application used by diners to order and pay for food service * Terminal app- the mobile/desktop application used by servers, kitchens, and owners to track, settle, and monitor sales for orders * Shall- indicates a must-have requirement * Should- indicates a nice-to-have requirement.   1. **Intended Audience** * Development team * Recruiters   1. **Scope** * Customer mobile app for ordering and payment * Business applications for accepting and tracking orders and payments * Integrate with existing payment system (Square) * Secure data transmission * Store business and transaction data with Square or independent server   System will improve the dining out experience by reducing wait times, contact, and eliminate the need to split checks by offloading tasks to a suite of mobile applications. This system will bring in revenue through product licenses while bolstering my resume with a demonstration of my skills while also advancing them. |

| 1. Scenarios |
| --- |
| * 1. **Product Perspective**   Oftentimes diners are anxious to finish their meal and pay when they have somewhere else to be, such as a movie or Broadway musical, or simply when they are tired or are on a bad date. They also choose to dine elsewhere when there is a long wait for a table but if they do remain, they may have trouble distancing from others.  Restaurateurs have an interest in turning tables to accommodate more paying guests but the added wait times for ordering and settling the bill constraint this considerably.  When dining in groups, there can be quite a circus around splitting the bill in communicating that to the server. This requires calculators, check splitting apps that are cumbersome, and often attempts to split shared items- all of which can lead to bad calculations and the headache of writing this out or hoping the server remembers the numbers when cashing diners out. Often it is simpler to just let the check and have; however, some diners order more items or more expensive items that the other party(ies) must subsidize.   * 1. **Product Features** * Diners shall be able to order and pay from their phone * Check splitting shall be mostly automatic * Groups shall link to a table across their personal devices by scanning a QR code displayed at the table   1. **Operating Environment**   In restaurants and bars, running alongside traditional POS systems while industry transitions to new modes of operation.   * 1. **Constraints** * Job search may reduce work hours available * Training may be necessary, slowing down development * May be called back to work before end of project * May be pulled away for unemployment activities * No Macintosh for iOS simulating * Limited number of devices for testing   1. **Assumptions and Dependencies** * Th Most people carry a smartphone * Most people are willing to learn a new system if it saves time and hassle * Most restaurants have WiFi and adequate cellular coverage * Access to the Square API * Encryption package |

| 1. Design overview |
| --- |
| * 1. **Logical View**   While the classes will be determined in development sprints, the states of the applications are fairly well defined. The diner app is diner-driven and has few states (fig. 4.1).  State diagram for user app  Fig. 4.1 – State Diagram: Diner app  The terminal app has three modes, depending on the user role. The server and kitchen spend a lot of idle time waiting for orders to come in and only change states when they need to take action. The admin mode is more user-driven for when an owner needs to review transactions, run reports, or settle transactions (fig. 4.2).  State diagram for admin app  Fig. 4.2 – State Diagram: Terminal app   * 1. **Process View**   The diner app shall have a standard login process where registration is required. Registration queries the Square API to check if the user already has payment details stored by Square (fig. 4.3). The user must have payment details on file in order to guarantee payment within the system.  The diner’s goal is to order food and pay through the app. Behind the scenes this is accomplished in a series of communications between the diner app, the cloud backend API, the terminal app, and the Square API (fig. 4.4).    Fig. 4.3 – Sequence Diagram: Login/Register    Fig. 4.4 – Sequence Diagram: Dine Out  Note: SetupOpenDinerPool has not been diagramed.   * 1. **Development View**   Both the diner app and the terminal app will run through the CMO API which will control data transmission to the CMO database and the Square API. The CMO database stores user and transaction information as well as menus while the Square API provides a payment gateway for the system.  The diner app has two main components consisting of a menu, downloaded through the CMO API, and a cart which is uploaded to the database.  The terminal app downloads information from the CMO the API in the form of packages known as tables. Tables consist of orders and transactions and may contain multiples of each if there are multiple diner app users in a group.  Component diagram  Fig. 4.5 – Component Diagram   * 1. **Physical View**   The system will run through four physical spaces consisting of the diner device (smartphone), the terminal device (smartphone, tablet, laptop), the CMO API server (application server), and the database server. Outside of the system exists another server hosting the Square API. The mobile applications will require local SQLite databases to temporarily hold data before submission.  Deployment diagram  Fig. 4.6 – Deployment Diagram |

| 1. Testing used |
| --- |
| * 1. **Development Testing**   << This section will be completed later in the project as the testing is completed. For now, the Test Plan will contain intended testing details. >>   * + 1. **Unit Testing**   Project will utilize test-driven development (TDD) methodologies during development. A CI pipeline will be created to trigger automated tests as changes are made to the code base.   * + 1. **Component Testing**   …   * + 1. **User Testing**   …   * 1. **Release Testing**      1. **Requirements Testing**   …   * + 1. **Scenario Testing**   …   * + 1. **Performance Testing**   …   * 1. **User Testing**      1. **Alpha Testing**   …   * + 1. **Beta Testing**   …   * + 1. **Acceptance Testing**   … |

| 5.0 DELIVERABLES | |
| --- | --- |
| **Name** | **Description** |
| Diner application | Mobile application for Android (iOS stubbing until testing available) |
| Terminal application | Mobile application for Android, UWP, x86 (iOS stubbing until testing available) |
| Application API | API to control data flow between applications and Square API |
| SQL database | Stores user and transaction data as well as menus |