Testing throughout the developmental stages of a new app or program is crucial to ensuring that the app not only runs as intended, but that it is meeting and preferably exceeding the needs and wants of its target market.

The testing strategy for the Bepoz app will involve a combination of formal and informal testing methods including unit testing, integration testing, system testing, exploratory testing and user feedback testing. Given the novelty of the app however, greater emphasis will be placed on exploratory and user feedback testing, as these are inherently the best methods for determining user friendliness and intuitiveness for users who have never used the app before. It is expected and hoped that improvements in features and functionality will be implemented as a result of feature requests and constructive criticism that arises throughout the testing process.

As our developers begin to create the product, testing of the absolute fundamentals will be performed in a process called unit testing. In programming, a ‘unit’ is the smallest testable part of a piece of software, and so unit testing is the act of running a block of code e.g a function, to ensure it works correctly (Unit Testing – Software Testing Fundamentals, 2020). Unit testing requires programming knowledge and is done usually at the time of writing the code, so the team’s developers Roshan and Simon will be responsible for performing this. Defects or bugs in the code will be corrected as they are found, negating any requirement for these to be formally documented or tracked.

Once the units of code have been compiled, integration testing can be performed by the team’s test analyst Joanna and support analyst Mason. Integration testing involves combining the individual components (units) of the software and testing them as a group. This helps identify any faults which may only become apparent when the components are made to interact with one another (Integration Testing – Software Testing Fundamentals, 2020).

In the context of our app, integration testing will consist of testing combined components which have direct relationships; for example, submitting an order and processing the credit card payment.

Once enough components of the app have been created that sequences of actions can be performed on a mocked up interface, some in-house system testing can occur. A checklist of items (Fig 1.) shall be used to ensure each component does get tested, and the same features will be tested at various intervals throughout the project’s lifetime to account for any inadvertent bugs that may appear due to changes elsewhere in the code. It’s at this point we can also draw upon the expertise of our industry based testers (Roshan’s family), to test the components and begin to provide feedback to the team in a collaborative fashion. Having this occur early on in the project is invaluable as industry experts will have insight on a multitude of variables specific to the restaurant environment and likely think of unique scenarios developers and testers simply could not. It is also far easier to make changes to the app earlier on and avoid the possibility of having to sacrifice many hours of work later on down the track when it’s discovered something needed to work differently.

When Bepoz has enough functionality for it to become a prototype, a broader range of testers will be sought to test the app in an explorative way. Explorative testing allows testers to have free reign of the app, and in no set order, explore and try whichever buttons and functions of the app they choose (Parmar, 2020). One of the key aims for the team is to create an app which any customer can easily understand and be able to use without assistance or prompts, and explorative testing is ideal in that it replicates this style of self learning behaviour.

Other factors relating to external variables will also need to be tested, including OS of the device that is being used, and performance of the system during heavy use periods (20+ concurrent users).

Fig 1. Testing Checklist

|  |  |  |
| --- | --- | --- |
| **Task** | Successful? Y or N | If not successful, why? Other comments/feedback |
| **Patron’s app** | | |
| Book a table |  |  |
| Scan QR code |  |  |
| Add items to order |  |  |
| Remove items from order |  |  |
| Make changes to items in order (e.g 2 serves instead of 1) |  |  |
| Place order and pay |  |  |
| Cancel order/flag waitstaff |  |  |
| Apply reward (applicable after 10th order) |  |  |
| Receive push notification from kitchen to collect food |  |  |
| Write a review |  |  |
|  |  |  |
| **Restaurant staff/Owner’s portal** | | |
| Order correctly prints on kitchen docket printer. Includes any custom notes/additions, allergies. |  |  |
| View reservations |  |  |
|  |  |  |
| View customer’s orders |  |  |
| Make amendments to orders – change size, add an allergy note, etc. |  |  |
| Admin are able to:   * Add items to menu * Remove items from menu * Temporarily make items unavailable * Make item available again * Add a special * Remove a special |  |  |

Our test users will preferably need to be a diverse group of people who represent the typical patronage in a restaurant setting, as well as those who are experienced in various roles at restaurants, including wait staff, kitchen staff, floor managers and business owners. Roshan’s inspiration for the app drew from a desire to help his family’s own restaurant, and so staff from the restaurant will be called upon in the early stages of testing. One to two waitstaff, a cook, and the business owner shall be sufficient to begin with. As testing increases in the later stages of the app’s development, a pool of roughly 15-20 testers of various ages and levels of technological savviness would be ideal, particularly to help replicate scenarios of heavy load/traffic on the system. The app could be offered as a free trial in the restaurant one evening, and/or beta testers could be sought via a company such as Ubertesters.

While it can never be known if a program is entirely bug free, some valid markers to indicate successful test phase completion would be an app which has had all known bugs corrected, and a consensus among the majority of users that the app would greatly benefit their business and workflow.

References

Parmar, D., 2020. *Exploratory Testing | Atlassian*. [online] Atlassian. Available at: <https://www.atlassian.com/continuous-delivery/software-testing/exploratory-testing> [Accessed 22 November 2020].

SOFTWARE TESTING Fundamentals. 2020. *Unit Testing - SOFTWARE TESTING Fundamentals*. [online] Available at: <https://softwaretestingfundamentals.com/unit-testing/> [Accessed 22 November 2020].

SOFTWARE TESTING Fundamentals. 2020. *Integration Testing - SOFTWARE TESTING Fundamentals*. [online] Available at: <https://softwaretestingfundamentals.com/integration-testing/> [Accessed 23 November 2020].