As the world becomes more connected than ever, people take for granted the ability to communicate to people around the world. With advancements in software and technology, software applications give us a platform to seamlessly communicate with whoever we want globally. The demand for scalable and resilient software applications is ever-growing. As applications evolve and user bases expand, transitioning from a local app to a distributed system becomes imperative to meet these demands. This paper outlines a comprehensive approach to transforming a local app into a distributed system, encompassing architectural considerations, technology choices, deployment strategies, and operational best practices.

Firstly, architectural considerations are important for the software system to function smoothly. Analyzing our architecture of the local app we made has made us decide to modularize certain components and microservices that can be extracted. The components we would modularize are our UI components, authorization/authentication for login, and our utility functions. Secondly, we would adopt a service-oriented architecture to decouple application logic into independently deployable services. This is important if we want to scale up our project. On top of that we would embrace event-driven design patterns to enable asynchronous communication between distributed components and improve scalability and responsiveness. Responsiveness is vital for having a good quality application that is distributed.

We would test our application with integration testing first to validate the interactions between distributed components and ensure interoperability and data consistency. After we run our integration testing we would then do resilient testing to make sure failure scenarios are tested

to assess the resilience and fault tolerance of the distributed system and identify potential weaknesses.

Our plan for deployment would first start with us choosing a cloud provider to host our large scale distributed application. For our project, we are going with Azure to host our product since we are familiar with that service. Since we are using Azure to host our application, we will use MYSQL for the distributed database because it comes with built-in features for scalability, availability, and global distribution. For our containerization, we have decided to use Docker over Kubernetes since we learned about Docker in class which is why we are more familiar with the service over Kubernetes.

In conclusion, transitioning a local app to a distributed system requires careful planning, strategic technology choices, and meticulous execution. By following the outlined approach and embracing the principles of scalability, reliability, and agility, we can successfully deploy our application so it can be considered in the realm of distributed computing.