Project Name: E- Park **Team Name:** OCM

Project Description:

E-Park is a mobile app that aims to alleviate the parking experience on campus. For students, faculty, and even just visitors, looking for an open parking spot, especially at peak hours can be cumbersome and time-consuming. E-Parc addresses this problem use a real time view of available parking, letting people view which parking zones have availability and how to navigate to an open park. With a simple and straightforward interface, E-Parc aims to take the anxiety out of parking and help users move from point A to point B, faster and with a little less stress.

Requirements Summary:

	Processor Cores	**OS**	**RAM**	**Storage**
MINIMUM REQUIREMENTS	**Dual core**	**Android 5.0 (Lollipop)**	**2 GB**	**300 – 400MB free space**
RECOMMENDED REQUIREMENTS	**Processor Cores**	**OS**	**RAM**	
	Quad core, Octa core	**Android 9.0 (Pie)**	**4GB and higher**	**600-700 MB free space**
OTHER REQUIREMENTS	**Permissions**	**Camera(for QR scan), Location, storage, notifications**		

Design Space:

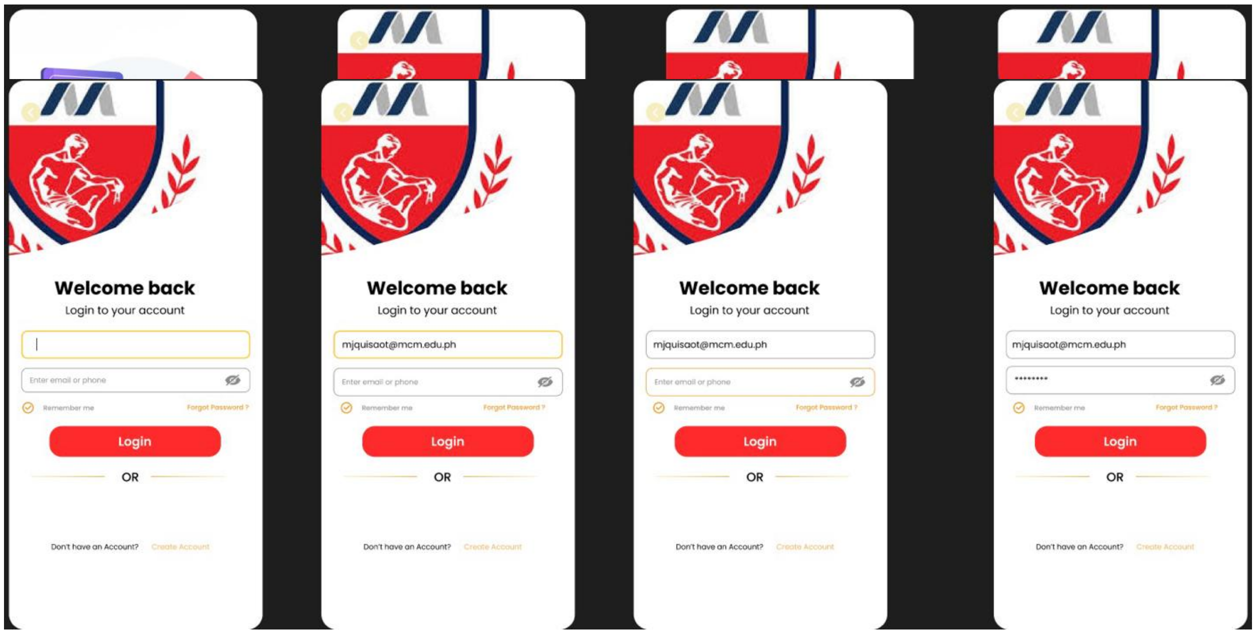
- **What requirements may be difficult to realize?**
- **What are some tradeoffs that you should or did explore?**
- **Which tasks will be easiest to support? Which tasks are the hardest?**

E-Park has limitations in providing accurate real-time parking spot availability since it will have to use new sensors, campus security systems and user reporting. Sensors also create cost issues (responsiveness), accuracy in the design process, although should the sensors clarify the positions of lost vehicles, they serve anticipatory functions. Responsiveness for an app, responds either within the app (local caching), or offline support for the app (long-term consequences), and thus an end-of-day system will also position how to maintain a real-time capacity. The equity of backward compatibility is also positioned around the possibility of the application running on older or new phones that may be outfitted with prior apps. Some of the limitations are clearly recognizable as there are tradeoffs in designing an app. If accuracy is needed, the sensor (costly) is your best option. Whenever predictive discourses are engaged (results updated automatically) it may not be at the same efficiency, and cost effective whenever machines are retrieved at the end of a day. Speed vs. detail is an issue, where if a service point, if speed were to be an on-demand service on the web is weighted into the detail you may have to make the burden (if for a moment) in the UI, or visually. Outputs from predictive algorithms, as a reference, the tribute displays are instantaneous. This would however carries a price unique to loading timelines. Battery and data could limit users, as features adept to real-time deliverables are also dependent on battery life and consumption, while weak network performance is a factor to restrict (or consume) possible data volume.

Design Summary:

E-Park is a parking app developed as a solution to the core problem of parking availability / user experience. The team originally considered the use of physical sensors to provide real time availability, however this option was found to be too expensive and complicated. As a result, they selected software making use of available institutional data or predictive models to find some balance between data availability, accuracy, and cost. The app also considered building a custom map service but decided again to go with third party GPS services from other established services (i.e., Google Maps) readily available within an established user interface. The design team discussed the use of either a fully featured interface or a simplified minimal interface, however eventually the team decided to use the minimal interface to simplify usability, load time and flexibility on devices. The team also discussed the complexities of a high-feature interface because of performance issues with older smart phones and low signal areas. The prototyping allowed the team to use Figma for collaboration, user flow, and simplicity in design. Figma also allowed all stakeholders to provide feedback and simple iterative design, as well involvement by multiple contributors.

Design:



Requirements Changes:

The developers for a parking app revised a lot of their original approach for real-time availability from exact spot-level availability to zone level availability. This opened up development for a more realistic approach as well as keeping core functionality. Their navigation interface relied on existing GPS applications such as Google Maps for the actual path taken to a parking zone, addressing user requirements with the development, instead of reinventing the wheel. We also revised the usability criteria after testing the high-fidelity prototype with early users. The focus was on simplicity and speed, particularly in high-pressure scenarios. A minimalist tap interface was desired to minimize user interactions to view availability and to start navigation. The previously understanding of offline support needed to be revised. We focused instead on graceful degradation, so the offline cached availability information could still be seen and the interface could still respond when users were in low signal areas. These adjustments came about organically as the developers transitioned from the planning phase to building a prototype in Figma. As they progressed through two distinct stages, they identified practical limitations, patterns of user behavior, and ways to slim down their app to provide sufficient value while being realistic about the entirety of what they could achieve within the boundaries and scope of the app.