List of possible exploits and attacks that can be executed on my system.

Attacks that can be performed on the web application layer

- Cross-Site Scripting (XSS):
    - For security measures and physical constrains that are attached to fingerprint scanners I will be creating a web app that runs on the local computer dedicated at the university premises, not only that but the new security measures that google and firefox have applied to chromium and gecko (browser engines for chrome and firefox) regarding cross-script cookies and resources it is impossible to do a cross-site scripting attack, the request would be rejected by the browser engine itself and return 400. Since July 14th of 2020 web servers are obligued to set the SameSite attribute and the respective CORS headers to handle preflight requests https://www.chromestatus.com/feature/5088147346030592
- NoSQL injection:
    - The web app does not communicate to the database directly, instead it performs a request to an intermediate server created with express-js that is being hosted on heroku which verifies the request and sends another request via the firebase admin-sdk which in turns calls Google cloud servers (realtime database) to perform CRUD on a nosql database; therefore, there is not any sql to work with to begin with, eliminating the possibility of a NoSQL injection attack.
- Path Traversal:
    - Since the application is being executed on a protected environment (university premises) it is virtually impossible to do a Path Traversal attack on that computer, unless you have unauthorized access to that computer. Apply security policies that ought to practise the idea of principle of least privileges.
- Local File Inclusion:
    - This is an issue which affects specifically web apps made with php that run on a server, since we are not using php to serve the contents and the web app is being executed on a protected environment, this attack is not possible.
- DDoS Attacks:
    - Unfortunately all web apps and its respective services are vulnerable to DDoS attacks, in our case we have two sservices where we can get a DDoS attack: Google Cloud and the intermediate cloud server that communicates with the Google Cloud, security is then handled by Google, in the case of the intermediate cloud server I will implement CORS protection and any additional IP protections directly on the code. https://firebase.google.com/docs/hosting/use-cases#access
- Command Injection:

    - As I said before, since the web application is being hosted in the university premises, we can avoid this attack by applying security policies that ought to practise the idea of principle of least privileges.

Attacks that can be performed on the mobile application layer

- Weak Server Side Controls:
    - This comes down to the security of the express-js application at the intermediate layer, I will be testing the routes throughly with sonarlint to identify any problems in the code.
- Lack Of Binary Protections:
    - Before deploying the application I will obsfucate the code to protect the binaries.
- Insecure Data Storage:
    - The mobile app will not use local storage for saving user settings, thus avoiding this problem entirely.
- Insufficient Transport Layer Protection:
    - I will be avoiding this attack by implementing SSL over HTTPS with lets encrypt when performing requests against my express-js application.

Availability of the cloud servers

Something I cannot avoid is the fact that like any cloud service, availability is not taken for granted, in the case of no availability an authentication cannot be performed, therefore the user will have to wait a few hours until the service is available again. One of the advantages of using the google cloud platform is that you can scale servers horizontally with help of the global CDN (Cloud Delivery Network), we can guarantee uptime of the platform as needed and scale thanks to the asynchronous nature of Ecmascript and NodeJs.