UML Diagrams Descriptions

Architecture Diagram:

Depicts the high-level view of the structure of the web application and how all its components interact with each other. We see that we integrate a CI/CD Pipeline using Azure which aids in automation and deployment thus linked to the backend and frontend components of the Web App. We have a Firebase database and Firebase authenticator. The firebase database is used and updated by the backend depending on what tasks need to be done such as registering a new user, blocking a user or creating an event and inserting it details in the events table. The web app offers a variety of functions/services that is handled by the backend while the information entered/selected is done so by the client via the frontend. The web app itself is hosted live on Azure.

Deployment Diagram:

Depicts the deployment architecture of the entire system and shows both the cloud-based components as well as the physical components. These include the Client PC with Web UI, the Azure Server with API server and GitHub page, and the Firebase Server which includes the firebase database and firebase authenticator. The diagram shows how all the components and devices link to and interact with each other across the different devices at a very high and abstracted level.

Class Diagram:

Depicts the system structure at a low level and describes how the entities/classes such as Users (Resident, Admin, Facility staff), Booking, Facility, Event, Reports, Identity Provider, and Maintenance Reports all interact and link to each other. It describes the methods and variables included in each class/entity and whether they are public or private as well as the parameters needed by the class when calling said methods. The diagram not only shows what entities/classes relate to what but also their cardinality restrictions.

Use Case Diagram:

Depicts the actors' (users of the system such as Resident, Facility Staff, Admin, as well as the authenticator) and the different use cases of the system. Describes how each actor may interact with the system performing different actions described by each use case. For instance, in the diagram we see that an admin can do a variety of things such as login, manage users, create events, manage bookings, and generate reports. All of which may include additional functionality such as extending to exporting reports for generate reports or sending notifications to all users when creating events. The diagram depicts all of this at a very high level with lots of abstraction so that nontechnical individuals such as stakeholders can still understand the overview of the system and who can do what when using it.

Component Diagram:

Depicts the structural organization of the software components, their interfaces, components, as well as their communication through exchanging of information to carry out their responsibilities. We see that all the services of the system are grouped together into an interface called Services which are not linked to each other as they are independent of one another, we see an Azure Cloud interface which houses the CI/CD pipeline, the backend API and frontend Web App of the system and how they, within the interface, link to each other and provide/receive from one another to carry out their responsibilities. The same goes for the Firebase interface. We then see how each interface connects to another in the flow of retrieving and providing information as well as if only a specific component within an interface is needed by another interface such as the backend API connecting to the Services interface.

Sequence Diagrams:

Sequence diagrams depict the interactions between the components, objects or actors over time in the system. Visualises the flow and order of exchanged messages to accomplish a specific task in each scenario. High level of abstraction and therefore easier for nontechnical individuals to understand.

Generate Reports – details the process for an admin to generate insightful reports that can be exported as a png, pdf, or csv file. The admin starts off by selecting the month and facility to thus applying a selection criterion for what they wish to generate reports for and then clicks the refresh all button to confirm. The frontend sends this to the backend which then queries the data given the required filtering by month and facility. The database receives this query and retrieves the desired data to the backend. The backend then formats this retrieved data and hands it to the frontend to display to the user in the form of tables and charts. The admin may then click the export button for the desired format type. The frontend requests export to the backend which then queries the data as earlier and formats and creates the exportable file which is then downloaded by the admin using the frontend.

Manage reports – Shows how admins and staff can interact with reports. The user (admin or facility staff) clicks on view reports button; the frontend requests the reports from the backend. The backend then queries to the database the reports. The database retrieves the data and hands it over to the backend which formats and displays the reports on the frontend. The user then attempts to update the report status. The frontend requests this from the backend which then is queried by the backend to the database (CRUD) which updates the status being stored in the database and returns a success message and then is displayed on the frontend by the backend stating so.

Reporting system – Shows the reporting system and retrieval. Resident enters a title and description and then selects the facility to be reported. The resident submits this via the fronted which requests to submit to the backend and so the backend queries this to the database to create a new report entry. The database does so and returns confirmation to the backend which the backend displays onto the frontend. The frontend then requests the reports and is queried by the backend to the database and is then retrieved by it. The backend then displays all the retrieved reports on the frontend for the user to view.

Schedule Event – Shows the process of an admin creating an event. User selects the type of event, the facility it will be hosted in, the start time, end time and date. The user then enters and title and description for the event and confirms. The frontend then submits the event request to thew backend which queries to update the event table in the database and validates it. The database then returns a success message to the backend which is then displayed on the frontend to reassure the user that their event has been successfully created.

User management – Shows how admins can edit actions and statuses of user (user control). The admin can view users in the system by selecting the button corresponding to it which then gets requested by the frontend to the backend which queries this to the database which the database then retrieves and is displayed by the backend on the frontend for the user to see. Similarly, the user can select approve user or assign them a role which then initiates a similar process to the viewing the users. For approving users, the request by the frontend is the same but the backend queries to update an existing user rather than retrieving all users from the database. For Assigning a role, the same process repeats but the query to the database by the backend is not an update rather than a retrieve. The user also has the option of revoking access/deleting a user from the system which involves sending a request via the frontend to the backend which then queries to delete the user or flag them depending on the request by the user. The database carries out the requested query and returns the success message to the backend which then formats a display message for the frontend so the user may gain reassurance their action was carried out successfully.

User Registration – Shows the process of registering a new user. The new user enters their required details (name, role) and then clicks sign in with google. The firebase authenticator then signs them in using their google account and creates unique user uid. The frontend then requests to POST the new user with their allocated uid using the backend using restful API. The backend then queries the user into the database to be stored and returns a success message or error message if something went wrong the frontend. This then tells the frontend where to redirect the user.

View Bookings – Shows the process of an admin/facility staff viewing, approving, and/or blocking a booking. The user (admin or facility staff) clicks on view bookings button which then triggers a request by the frontend to the backend for the bookings. The backend then queries to fetch all bookings stored in the database and so the database returns the bookings and all their associated information to the backend which is then formatted by the backend and displayed on the frontend for the user to view. The user can then approve or block the booking they select from the list (in a table). The frontend then sends the appropriate request to the backend (approve or block). The backend queries this to the database and the database then changes the status of the booking in the bookings table and returns a success message to the backend which is then displayed on the frontend to the user.

Booking System – Shows the process of a resident creating a booking. The user (resident) selects a facility and day for their booking. This triggers a request to be made by the frontend to the backend. The backend then queries to retrieve all available booking slots in the database. The database then retrieves the available slots and is displayed on the frontend by the backend. The user then selects from the available slots with the number of people and timeslot. This is then sent by the frontend to the backend which creates a pending booking request and returns a success message to the frontend and is displayed to the user. On the other end of the system, the admin can carry out the process explained in view bookings sequence diagram. Once done, the frontend notifies the user of the approved/blocked booking.