Architecture

This web application architecture is currently composed of three python files. The three files are frontend.py, backend.py, and models.py. Along with these three python files, we have testing files that will test the frontend, and integration. We also have four .html files that make up the frontend of this web application. Now we will investigate the architecture of each of the python files with an overview of each class, and each function.

Frontend

The frontend python file is composed of a variety of functions which will deal with different routes. Please refer to the below table depicting the function name, route, input, output, and purpose.

Function Name	Route	Input	Output	Purpose
`register_get()`	`/register`	None	Returns the rendered template register.html file with an empty message.	The purpose of this function is to handle routing for registering a user when they first make a request to the /register route. This will ensure they are represented with the register.html page to register their account.
`register_post()`	`/register`	None	Returns either the register.html file with an error message if a registration error occurred, otherwise redirects to /login route	The purpose of this function is to handle the /register route for POST, which is when the user actually registers themself a new account and the account is added to the database.
`login_get()`	`/login`	None	Returns the rendered template login.html file with a message that reads 'Please login'.	The purpose of this function is to handle the /login route for GET method. It will render the login.html file with the message "Please login"

login_post()	/login	None	Returns either the login.html file with an error message if a login error occurred in regards to credentials,or if there is an error logging the user in then returns a rendered template of the login.html file with a message that reads 'login failed', or a redirection to the / route with code 303 if no errors	The purpose of this function is to handle the /login route for POST method. It checks for validity of email and password, informing the user if they input invalid credentials. If the credentials are valid, the user is logged in. If they aren't, then the program will inform the user that their credentials are incorrect.
logout()	/logout	None	Returns a redirection to the / route, if the user was already logged in.	The purpose of this function is to handle the /logout route. If the user is logged, then this method will log them out and redirect them to the / route
authenticate(inner_function)		Takes any python function that accepts a user object	Returns whatever is returned from the inner function with the user object passed to it if the user is logged in, otherwise returns a redirection to the / route.	The purpose of this function is to authenticate the user for functions that use a User object.
profile(user)	/	User object which holds all relevant user	Returns a rendered template of the index.html	The purpose of this function is to handle the / route. The user is first

		information	file with the user object set as the user and the ticket object set as the ticket.	authenticated, and then sent to their profile page.
error()	/*	None	Returns a redirection to the / route with an error code 404	The purpose of this function is to handle any route not defined above. This method will return a 404 error code, and redirect the user to the / route.

Backend Model

The backend python file is composed of a variety of functions which will deal with different requests to the database. Please refer to the below table depicting the function name, input, output, and purpose.

Function Name	Input	Output	Purpose
get_user(email)	This function takes the users email address as a parameter	This function will return an instance of the user object returned from the SQL database	The purpose of this function is to get a user object given their email.
login_user(email, password)	This function takes the users email and password as parameters	This function will return an instance of the user object if the user is successfully logged in	The purpose of this function is to check the user authentication by comparing the password entered with the password supplied
register_user(email, name, password, password2)	This function takes the users email, password, name, and second password as	This function will return None	The purpose of this function is to register a user into the SQL database

	parameters		
get_all_tickets()	This function takes	This function will return a list of all of the tickets	The purpose of this function is to return a list of all tickets that belong to a given user.

Model

The model python file is composed of two classes which will act as the models for this application. Please refer to the below table depicting the class name, input, output, and purpose.

Class Name	Input	Output	Purpose
User(db.Model)	This function takes the SQL database as a parameter	This class does not produce an output	The purpose of this class is to establish the attributes of a User object, and input those attributes into the database
Ticket(db.Model)	This function takes the SQL database as a parameter	This class does not produce an output	The purpose of this class is to establish the attributes of a Ticket object, and input those attributes into the database

Solution Structure

The three python files described in the architecture make up the solution structure.
The models file will hold the models for the objects used in the application. The current objects described in models.py are the User object and the Ticket object. The User object is created by a client when they first register an account. The registration of an account starts with the user entering their credentials on the /register page. If their credentials fit the requirements, which is checked by the register_post() function in the fronted.py file, then the credentials will be sent to

backend.py in order to store the user in the database. The user can log back in with their credentials, which is checked by the login_post() function in frontend.py. The backend.py file will authenticate the login, and then retrieve their User object from the SQL database.