Travel Insurance Administration Application

Client: Anonymous
Business Size: Corporation
Industry: Insurance and Risk Management
Country: UK
Technology: ASP.Net Core 6 MVC, Razor Pages and Entity Framework Core, C#, SQL Server, Azure DevOps

Objective: Replace and update existing travel insurance administration application

The Brief

The client had a web application to manage group travel administration for higher education establishments in the UK which did not conform to a new set of standards required for the storage of users' identities for authentication and authorization and the decision had been made to replace it.

Background

The client is a global insurance brokerage, risk management, and HR & benefits consulting company. As part of its services, they offer a web application to their UK university client base to record travel arrangements and validation of insurance cover for students and lecturers. The application allows the user to download a certificate as proof of travel insurance approval.

Methodology

The project utilized the Agile framework with two-week sprints, daily stand-up meetings, sprint planning, backlog refinement, sprint review, and retrospective sessions. Various tools were used, including Visual Studio 2022, SQL Server Management Studio, JIRA, and MS Teams. Microsoft Asp.net Core 6 MVC with Identity and Entity Framework Core were used for the software solution's development. Other technology stacks used were Sitecore and Sitecore API, Gulp, and Figma. Bitbucket was used for source control and Azure DevOps for the build and release pipeline into the QA environment.

The replacement web application would be managed by a support team, as well as their respective UK university customers admin staff, with access controlled by a mixture of Windows credentials for support staff, and application specific logins for students and lecturers.

The new web site can be broken down into three core functionalities:
- Authentication and Authorization management
- Travel and certification proof of insurance management
- Administration of the system

Authentication and Authorization management – Using Microsoft ASP.Net Core 6 MVC with Identity. The application referenced the in-built libraries from 'Identity', which have methods that are called to managed the registration, account creation, customer's profile, login, password, two-form authentication with one-time password, and the confirmation and validation processes. It also manages users' role-based authorization to control access to application features. In terms of security, 'Identity' also provides methods to generate the appropriate encrypted tokens that are passed into emails sent to the user for action regarding confirming their email, password reset and one-time password on two form authentication process.

The validation process checks the user's email domain against a separately managed central list containing their admin staff email, their domain email, their set of specific questions, and their certificate as proof of approved insurance to travel.

Travel and certification proof of insurance management – The application records each user's travelling start and end dates, and trip destination details. It allows them to add more trips and edit existing trips. A summary page is displayed where the user can see their trips and can download a certificate for each trip.

Administration of the system – This part of the system allows the client's support staff and their respective UK university admin staff to deal with the administration core functionality. For this phase our involvement was limited to setting up role-based authorization so that

the client's developers could create the relevant admin pages to define what the users and admin staff can access.

Challenges
As a team of four developers (two from the client and two from OCS Consulting) our initial challenge was to setup Visual Studio 2022, SQL Server Developer's edition, Gulp, and access to the Sitecore and Figma Design applications before starting any development work. Security issues made this less than straightforward.

We also needed to incorporate the ASP.Net core Identity into the Visual Studio project, which builds the Identity tables in the database. This was distributed within the team via Development and QA branches in Bitbucket for the rest of the developers to clone.

Another challenge we had was Gulp was not installing properly. This was needed to compile the SCSS file, but only one developer from the client was able to use Gulp on their laptop so the team was reliant on that developer to compile our SCSS files when creating new CSS rules to style the web pages.

There were a lot of complex business rules for the trips, profile, emergency contacts, login, registration, forgot password, and user and date of birth validation. Custom classes needed to be created for some of these validation requirements, as well as for token life span.

To resolve a SQL Server database issue, a SQL Server database was created in the development environment, allowing development work to continue as it required DDL work, such as adding new fields for Registration, Accounts, Profile, and creating a new table for Emergency Contact.

Our consultant researched these obstacles and overcame them to complete the project tasks assigned successfully and on time.

Consultant Contribution
Our consultant was responsible for developing all the Identity core functionalities for the application. This involved creating Razor pages where the model and controller code are incorporated into the web

pages for Registration, Accounts, Login, Forgot Password, User's login and date of birth validation, Reset Password, Email Account Confirmation, Account Locked out confirmation, and two-form authentication with one-time password. Razor was also utilized for the Email component and Email Templates (and their HTML and CSS styling code) that send emails with the relevant data and encrypted tokens, for email confirmation, account locked out, and one-time password.

Application changes included the incorporation of Authentication and Authorization functionality so that users cannot enter web pages via URL directly. Data passed into the URL as route parameter values is encrypted to avoid any meaningful data being visually displayed.

Other developments included MVC pages to manage the users' profiles, emergency contacts, and password recovery.

Every development followed the business requirements for security and styling according to the client's style template.

During migration to the new application, modifications to the database tables were made as part of the migration process to support the modernized security regime, with the client's administrators being pre-configured with suitable access rights.

Lessons learned

The people involved were friendly and approachable, effectively allowing shared ownership of this project. This also facilitated understanding and consideration of each-other's challenges, and collaboration with effective communication. As a result, challenging issues were dealt with and a successful software solution was delivered.

Our consultant, who was initially expected to just write the code as per the client's requirements, ended up collaborating with the business analyst and testers as well as working with the other developers in the team.