Executive Summary

- PCD's development has to date been entirely outsourced to a third party provider, with no technology experience within the PCD team. However, Coax, the 3rd party provider, has developed a system without any significant technical flaws. The software stack used is standard and visual checks of the codebase shows it is well layered and it was easy to follow.
- There are opportunities for improvements in the user experience and the ability to increase engagement. The methods to generate buyer and seller traffic are areas that needs most urgent attention.
- Currently there is little traffic, and the business needs a strategy here. Once the traffic increases, continual user experience work will be required to convert visitors to paying and active customers.
- As traffic increases, the business will also need more functionality on the CRM side to help them drive the business and see insights.
- Market requirements and user experience analysis may also lead to a need for more functionality in the application.
- Upgrades around ticketing, testing and documentation are important but not right now as generating the traffic, fixing any UI flaws and turning visitors into customers is a priority.

Overview

Activity	RAG	Observations	Considerations	Recommendations
Overview		• PCD development started in Feb 2020. It started with the UX/UI development, with the full application development started soon after • Development has been done solely by a third party development company Coax, who are based in Ukraine • In the beginning there were 4 people on the project. Currently development is on hold, with the last large activity done in summer of 2022 • There is currently a part time team of 2 who looks after the maintenance of the project. One full stack developer who is the project lead, and one infrastructure support person • System is hosted on AWS, with 2 virtual machines used for production. The system also requires a Postgres database, an ElasticSearch search engine and a Redis cache, these are provided by AWS services • The software code base is written in Javascript and is held on Github. Github gives it version control as well as deployment control triggers	• The market identification and the requirements came from the founder, there is no internal experienced technology person. The technological direction is left entirely to Coax • There is no visibility of forward planning for development activities	• A development plan based on the business strategy, going out at least 3 months will help the business understand what is and will be possible. Will also help in resource planning

System and software

Activity	RAG	Observations	Considerations	Recommendations
System architecture		• PCD systems are hosted entirely on AWS. There are 2 servers used for production, one frontend and one backend. There are 4 other servers used for development and testing • Data is stored in a Postgres database, this is provided by the AWS RDS service • Search is provided by an ElasticSearch this is provided by the AWS OpenSearch service • There is also a Redis data cache, this is provided by the AWS ElastiCache service	• This setup of is fairly common for projects of this type • The use of AWS services where it can is a good choice • The 2 virtual machines are of quite low specification. If the traffic were to grow this will need revisiting as to how best to scale, be it increasing the instance specification or scaling to multiple servers	• Investigate to see if the software stack both frontend and backend are also capable of running in a serverless environment. This will allow the use of services instead of the 2 virtual machines. This will make the issue of scaling a little easier going forward and deployment simpler
Software		• Javascript is used throughout the system. There is a React based frontend and a Node.js based backend • There are 2 Travelport microservices, also written in Javascript that does the hotels API connections. These work in the backend servers too • SSR (Server Side Render) is used in the frontend to help handle SEO properly • The Ant Design framework is used to provide the front end look and feel • Sequelize is used as the ORM tool, this makes it easier to map the internal objects to database tables and also helps in writing the database queries	• This setup of is fairly common for projects of this type • Javascript is a good choice as is the React framework for frontend and the Node.js backend. • Upon visual inspection of the code it was fairly easy to work out what various parts are doing • The use of And Design and Sequelize are also good choices and will help minimise the amount of coding	• None

Development

Activity	RAG	Observations	Considerations	Recommendations
Development		• PCD's system is entirely coded and managed by Coax • Github is used as the code repository, there are 4 projects within the repository. Broken up between frontend, backend and then there are 2 projects for microservices that works with the external APIs • Developers work locally on their own PCs and merge their work into the repository when ready	• The development process is a standard one and the use of Github is also standard	• None
Ticketing		• Jira is used internally by Coax, PCD has no visibility of this • At the moment, issues are reported via Slack and the Coax project manager creates the issue in Jira	• Ticketing is used and managed internally by Coax	• A ticketing system used and managed by PCD would make interrogation of changes, timeline management and assessment of forward resource requirements easier
Payments		• Payments is made via MangoPay • If it is a prepaid listing the payment url from MangoPay is used, the user is just redirected to that url • If it is payment at the property - the credit card details is gathered and sent to MangoPay. With this way, credit card details is sent to our server and then forwarded to MangoPay	• MangoPay does not handle transactions above £1,990 in the UK • MangoPay recommends using a direct connection to them when collecting credit card details. It is not best practice to have this traffic going through own servers even when not saving to the database	• Using another provider is under investigation • To continue with MangoPay the collection of credit card details may need revisiting. Apparently this process has not been used much, just need to clarify its need.

Testing and documentation

Activity	RAG	Observations	Considerations	Recommendations
Testing		• There are some unit testing software across the codebase. These are run automatically pre-deployment to both staging and production • Functional testing is done manually • SonarQube is used for static code analysis	• The backend appears quite well covered by tests, with tests for majority of functions within the api • The frontend appears lacking in tests, there are some tests around the React pages but they appear quite simplistic • There is no automated functional testing	• More automated testing scripts should be created for the frontend • Automated functional testing scripts should be created - using something like Storybook, which also helps in frontend development a set of functional tests can be created
Load testing		• This has been carried out historically via the use of external tool	• Currently the load performance is unknown, it seems OK but the instances are quite small so may require upgrading	• Load testing should become a regular activity as changes may lead to areas of concern that has not arisen before
Penetration testing		• This has not been carried out		• May not be appropriate at this point in the life of the application, but some penetration testing by a third party should be done as the traffic scales
Documentation		• This is no documentation giving an overview of the system and how it all works • There are some readme files in the codebase, but there is little documentation otherwise	• A set of high level documentation provides a good introduction to the project for both the business and developers; it also gives an insight into why certain decisions were taken	• A set of high level documentation is a minimum. It does not need to be excessive but a process to encourage the developers to put some documentation down for each module would be helpful

Deployment

Activity	RAG	Observations	Considerations	Recommendations
Deployment		• Deployment is done via committing to a Github branch. The CI/CD (continuous integration / continuous deployment) process runs once a commit is done on the repository, then the tests are run which generate reports • Also the code is pushed to the COAX SonarQube server for static code analysis • Then for staging and pre-prod, using AWS SSM Run Command, the deploy.sh script on the remote servers are run (either staging or pre-prod) • For production, GH repository invokes AWS CodePipeline which launches AWS CodeBuild. CodeBuild - build docker images and push them to the AWS ECR repository. Then manually run deploy.sh script for rebuilding the docker containers application (this step can be automated)	• This is a standard process	• There are some parts which can be further automated. Not important right now, but something that should be done as the system attracts more users and requires further enhancements • Staging should mimic production as much as possible and that includes the deployment process

Application

Activity	RAG	Observations	Considerations	Recommendations
SEO		• Google analytics is setup to monitor the traffic and its details	• There is too little traffic to draw any major conclusions at the moment	• More traffic needs to come first. Beyond that more can be done with tags to further manage where people have been
User Experience		• There is no user experience tool present	• A user experience tool, such as Fullstory, will help analyse the user experience when they are on the site. It helps to see when people are dropping out and potentially why	• Use a user experience tool, such as Fullstory
PageSpeed Insights		• PageSpeed Insights is a free Google tool that measures the performance of pages	• In mobile view the blog page only scores 32 for performance. This is due to the Javascript size and the large main image	• Investigate doing code-splitting as blog pages do not require the app functions • Investigate image size reduction or preloading images
Blog		• The blog system was developed in house by Coax • The page entry uses Quill a popular open source editing tool	• System seems to contain all the elements they require	• None
CRM		• The CRM was developed in the admin to give visibility of the data in the system	• Currently it serves its purpose, but more thought is required as to what KPIs and reporting is required, and if there are any missing data	• Some design is required as to future direction of the CRM

Development and code assessment

Development Team and Resources

Coax appear to have a large and capable team. There is no in house person with deep technology understanding

SDEP - Version Control

Github is used

SDEP – Static Code Analysis

Sonarqube is used

SDEP- Manual Code Review

Currently not relevant as only one person is doing development

SDEP – Testing

There are some tests, more in the backend than frontend

SDEP – Source Code Documentation

The source code filenames, function names and variable names are all named according to their use. Code is well laid out and easy to follow

SDEP – Automated build and deploy

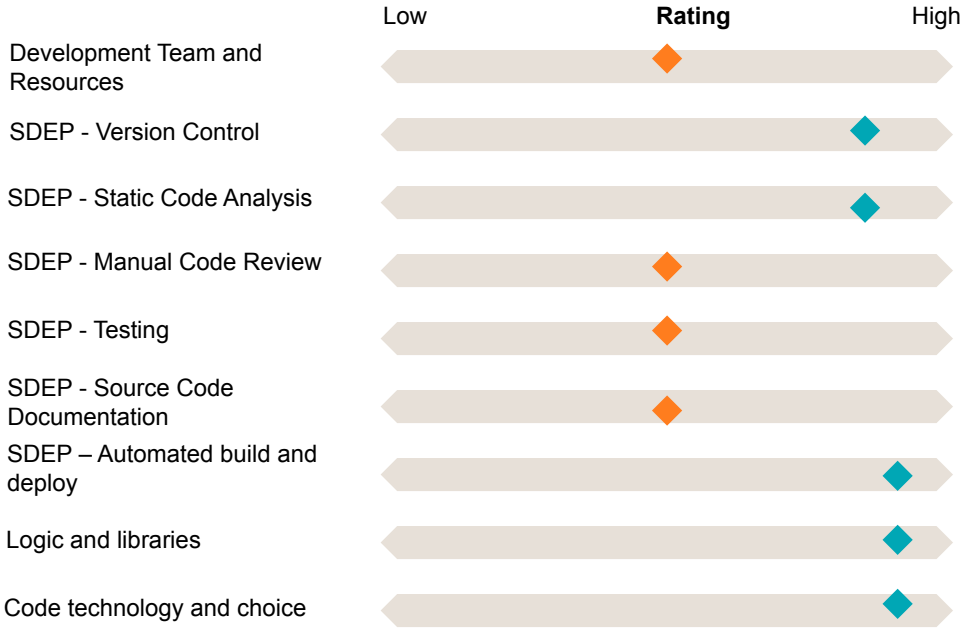
Done via AWS CodePipeline and CodeBuild

Logic and libraries

3rd Party libraries are well managed and does not appear excessive

Core Technology and choice

Software stack used a good choice for this type of application



Development and code assessment (cont.)

Project Management

The project appears to have been well managed

Metrics – Code Complexity

The code did not appear to be highly complex and was well laid out for reading and understanding from our perspective

Metrics – Code Duplication

There is little evidence of code duplication, the layers and the functions are well laid out

Metrics - Coupling

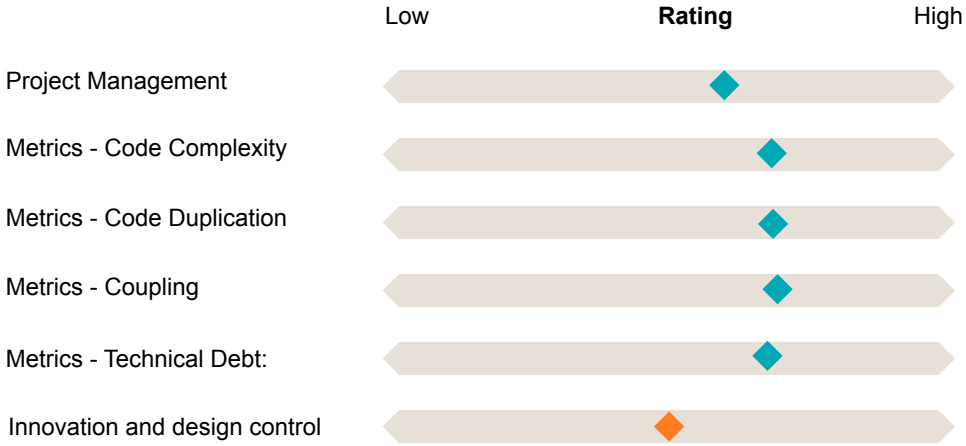
The coding is separated clearly to layers and they are well decoupled

Metrics Technical Debt

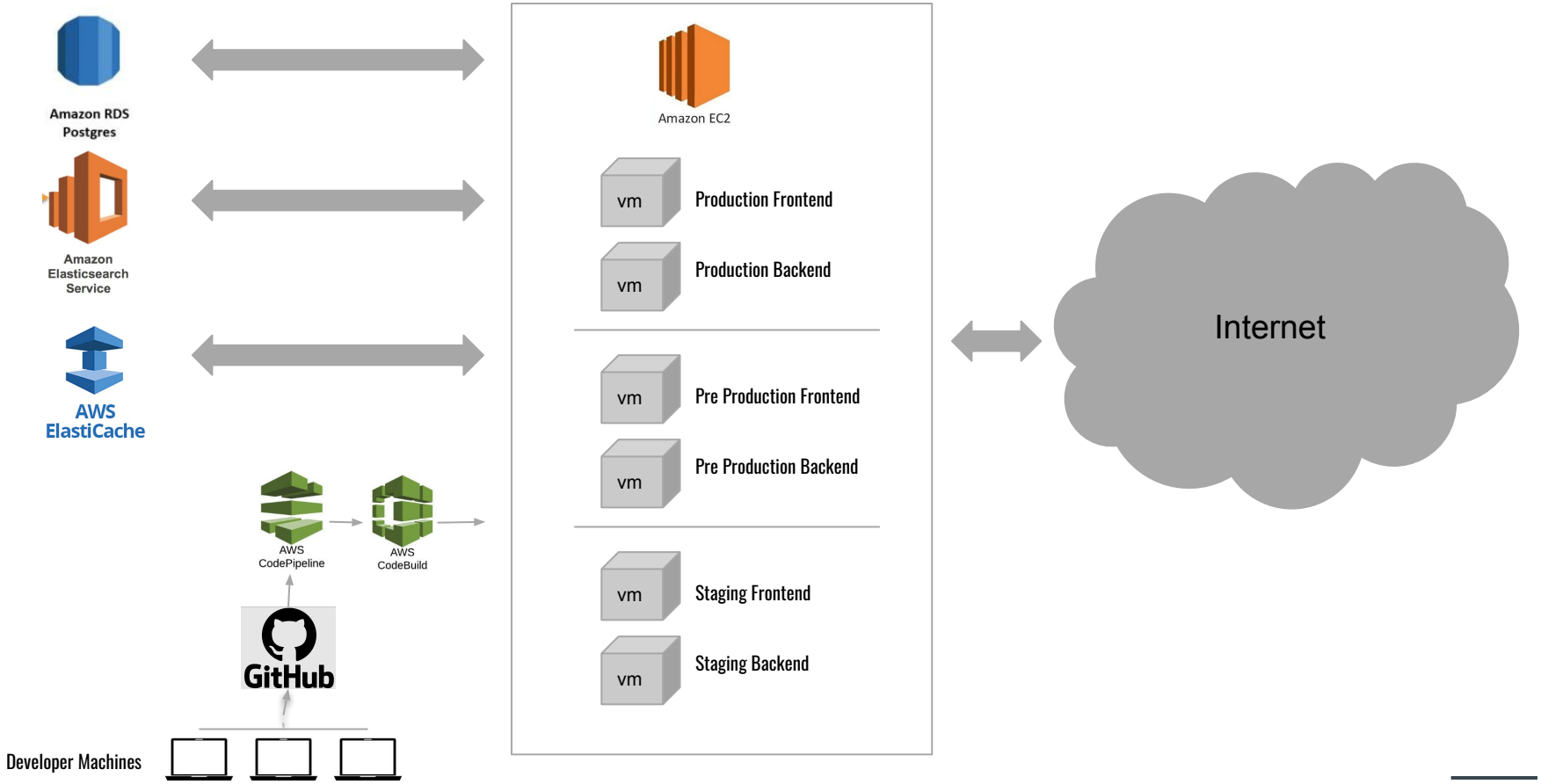
Nothing significant

Innovation and design control

There does not appear to have been any significant need for innovation or development of new approaches to achieve the desired functionality



Development flow and server architecture



Next steps

- The next 3 months should involve the transition to a new technology ownership and the creation of sprints or workstreams to work on the current priority items
- A technical lead and an on demand (full stack) developer resource should cover this period
- During this time and subject to available funds the SEO attack plan and strategic partners/ Hotel groups onboarding methods can also start to develop
- The integrations with hotel owners and groups will need an API with a view to develop the existing travelport method
- As the the brand becomes more synonymous with resale of bookings the product can become more prominent in the site. See https://www.airbnb.co.uk
- Customers buy with their eyes!
- All account info on the platforms that PCD use should have email contacts of a technical lead on updates and software releases.
- All support and technical issues should be documented to create a knowledge pool so the business is never reliant on any individual
- All future features and product development should be done via a story book type product https://storybook.js.org