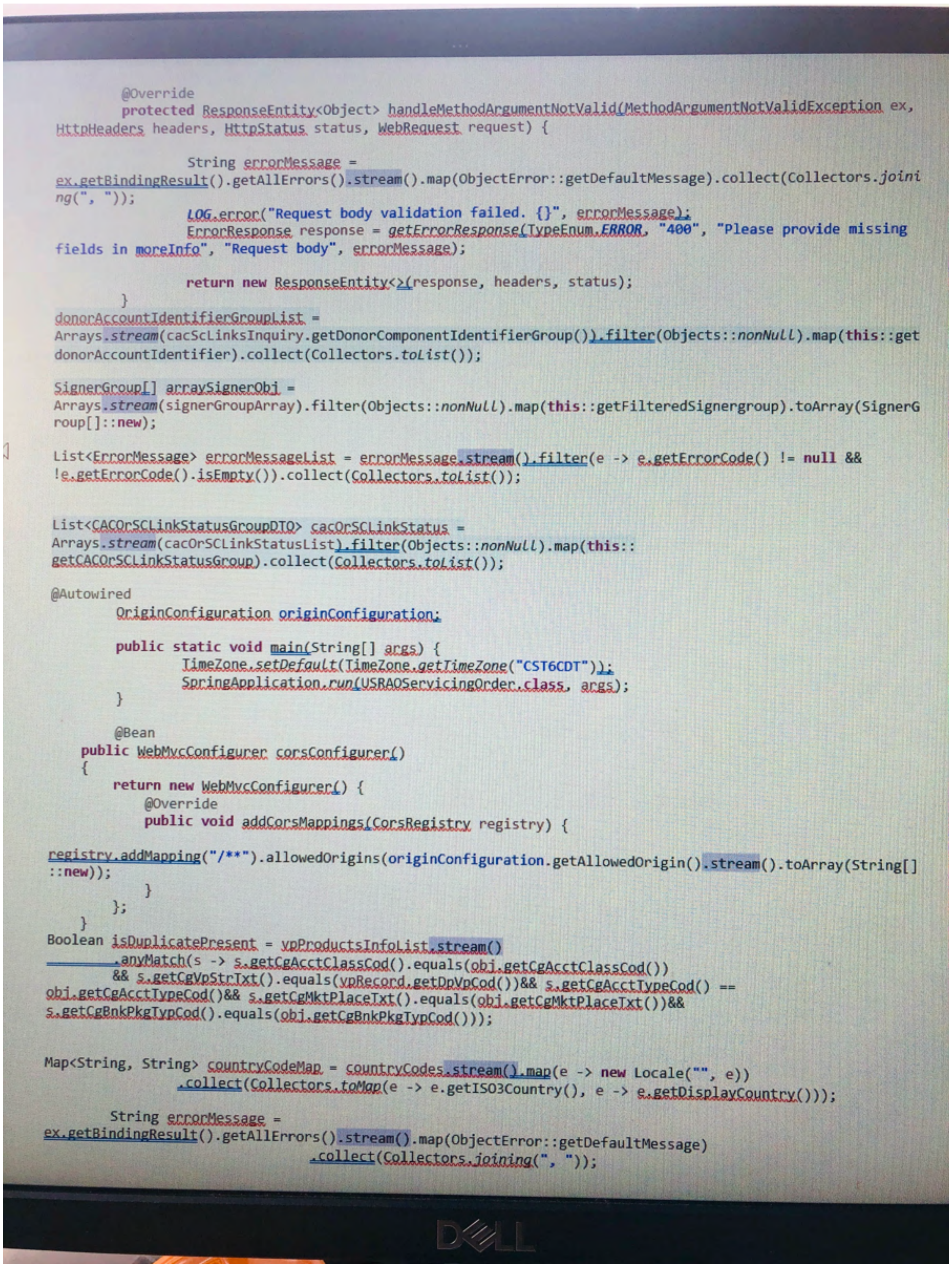
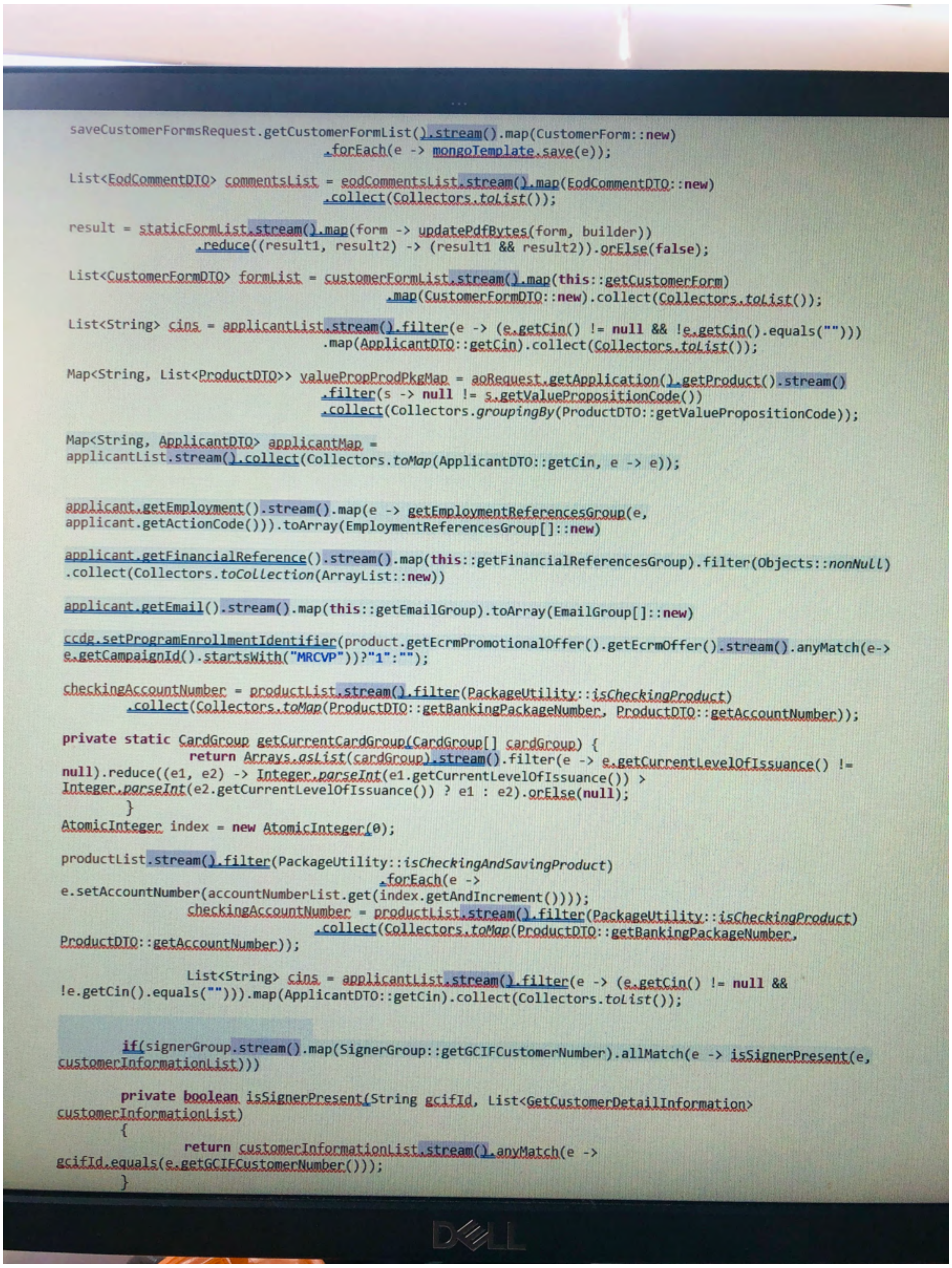
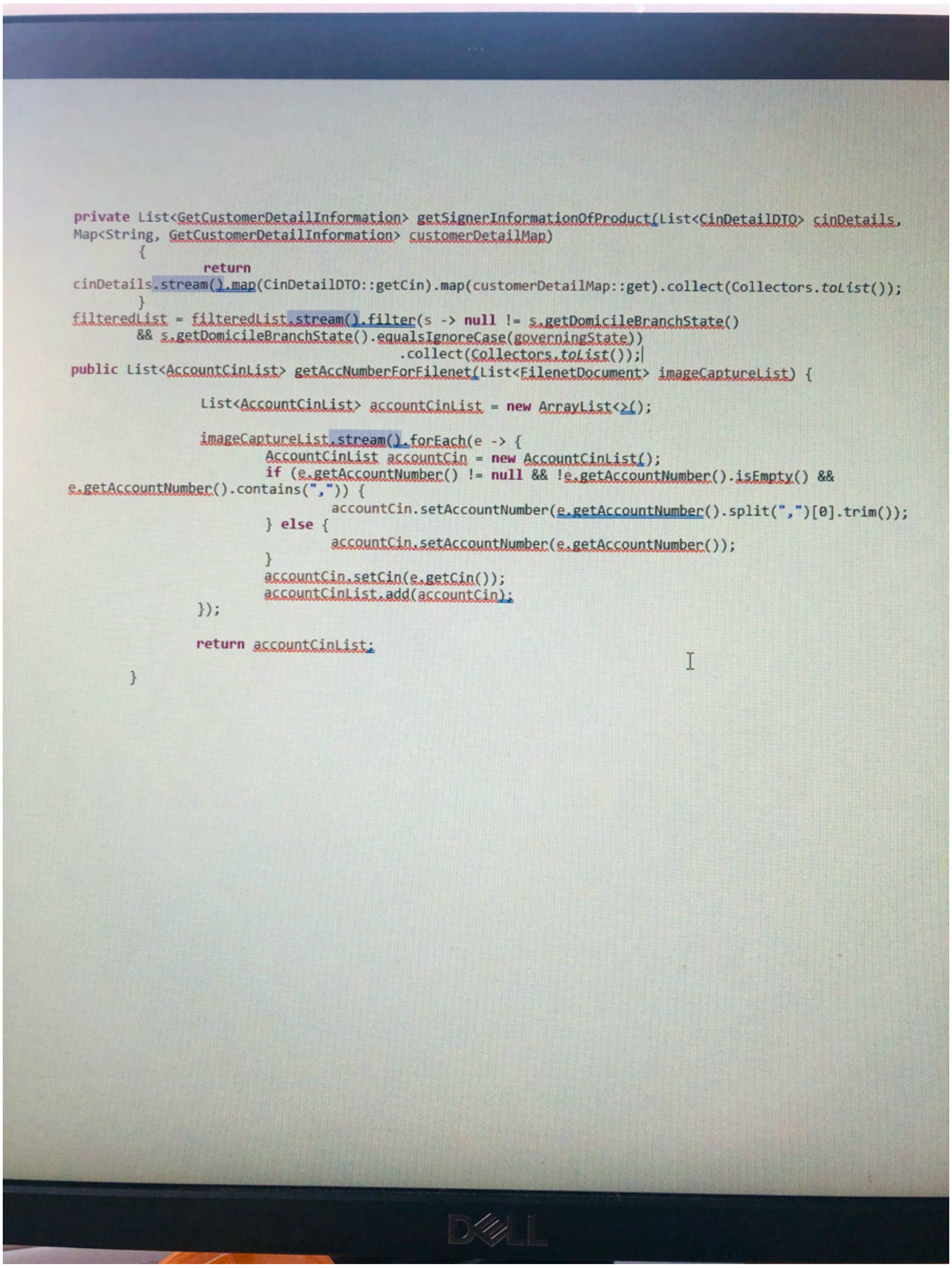


```java
        @Override
        protected ResponseEntity<Object> handleMethodArgumentNotValid(MethodArgumentNotValidException ex,
HttpHeaders headers, HttpStatus status, WebRequest request) {

                String errorMessage =
ex.getBindingResult().getAllErrors().stream().map(ObjectError::getDefaultMessage).collect(Collectors.joini
ng(", "));
                LOG.error("Request body validation failed. {}", errorMessage);
                ErrorResponse response = getErrorResponse(TypeEnum.ERROR, "400", "Please provide missing
fields in moreInfo", "Request body", errorMessage);

                return new ResponseEntity<>(response, headers, status);
        }
donorAccountIdentifierGroupList =
Arrays.stream(cacScLinksInquiry.getDonorComponentIdentifierGroup()).filter(Objects::nonNull).map(this::get
donorAccountIdentifier).collect(Collectors.toList());

SignerGroup[] arraySignerObj =
Arrays.stream(signerGroupArray).filter(Objects::nonNull).map(this::getFilteredSignergroup).toArray(SignerG
roup[]::new);

List<ErrorMessage> errorMessageList = errorMessage.stream().filter(e -> e.getErrorCode() != null &&
!e.getErrorCode().isEmpty()).collect(Collectors.toList());


List<CACOrSCLinkStatusGroupDTO> cacOrSCLinkStatus =
Arrays.stream(cacOrSCLinkStatusList).filter(Objects::nonNull).map(this::
getCACOrSCLinkStatusGroup).collect(Collectors.toList());

@Autowired
        OriginConfiguration originConfiguration;

        public static void main(String[] args) {
                TimeZone.setDefault(TimeZone.getTimeZone("CST6CDT"));
                SpringApplication.run(USRAOServicingOrder.class, args);
        }

        @Bean
    public WebMvcConfigurer corsConfigurer()
    {
        return new WebMvcConfigurer() {
            @Override
            public void addCorsMappings(CorsRegistry registry) {

registry.addMapping("/**").allowedOrigins(originConfiguration.getAllowedOrigin().stream().toArray(String[]
::new));
            }
        };
    }
Boolean isDuplicatePresent = vpProductsInfoList.stream()
                .anyMatch(s -> s.getCgAcctClassCod().equals(obj.getCgAcctClassCod())
        && s.getCgVpStrTxt().equals(vpRecord.getDpVpCod())&& s.getCgAcctTypeCod() ==
obj.getCgAcctTypeCod()&& s.getCgMktPlaceTxt().equals(obj.getCgMktPlaceTxt())&&
s.getCgBnkPkgTypCod().equals(obj.getCgBnkPkgTypCod()));


Map<String, String> countryCodeMap = countryCodes.stream().map(e -> new Locale("", e))
                .collect(Collectors.toMap(e -> e.getISO3Country(), e -> e.getDisplayCountry()));

        String errorMessage =
ex.getBindingResult().getAllErrors().stream().map(ObjectError::getDefaultMessage)
                                .collect(Collectors.joining(", "));
```



```java
saveCustomerFormsRequest.getCustomerFormList().stream().map(CustomerForm::new)
                                       .forEach(e -> mongoTemplate.save(e));

List<EodCommentDTO> commentsList = eodCommentsList.stream().map(EodCommentDTO::new)
                                       .collect(Collectors.toList());

result = staticFormList.stream().map(form -> updatePdfBytes(form, builder))
                   .reduce((result1, result2) -> (result1 && result2)).orElse(false);

List<CustomerFormDTO> formList = customerFormList.stream().map(this::getCustomerForm)
                                                .map(CustomerFormDTO::new).collect(Collectors.toList());

List<String> cins = applicantList.stream().filter(e -> (e.getCin() != null && !e.getCin().equals("")))
                                       .map(ApplicantDTO::getCin).collect(Collectors.toList());

Map<String, List<ProductDTO>> valuePropProdPkgMap = aoRequest.getApplication().getProduct().stream()
                                       .filter(s -> null != s.getValuePropositionCode())
                                       .collect(Collectors.groupingBy(ProductDTO::getValuePropositionCode));

Map<String, ApplicantDTO> applicantMap =
applicantList.stream().collect(Collectors.toMap(ApplicantDTO::getCin, e -> e));


applicant.getEmployment().stream().map(e -> getEmploymentReferencesGroup(e,
applicant.getActionCode())).toArray(EmploymentReferencesGroup[]::new)

applicant.getFinancialReference().stream().map(this::getFinancialReferencesGroup).filter(Objects::nonNull)
.collect(Collectors.toCollection(ArrayList::new))

applicant.getEmail().stream().map(this::getEmailGroup).toArray(EmailGroup[]::new)

ccdg.setProgramEnrollmentIdentifier(product.getEcrmPromotionalOffer().getEcrmOffer().stream().anyMatch(e->
e.getCampaignId().startsWith("MRCVP"))?"1":"");

checkingAccountNumber = productList.stream().filter(PackageUtility::isCheckingProduct)
        .collect(Collectors.toMap(ProductDTO::getBankingPackageNumber, ProductDTO::getAccountNumber));

private static CardGroup getCurrentCardGroup(CardGroup[] cardGroup) {
                return Arrays.asList(cardGroup).stream().filter(e -> e.getCurrentLevelOfIssuance() !=
null).reduce((e1, e2) -> Integer.parseInt(e1.getCurrentLevelOfIssuance()) >
Integer.parseInt(e2.getCurrentLevelOfIssuance()) ? e1 : e2).orElse(null);
        }
AtomicInteger index = new AtomicInteger(0);

productList.stream().filter(PackageUtility::isCheckingAndSavingProduct)
                                             .forEach(e ->
e.setAccountNumber(accountNumberList.get(index.getAndIncrement())));
                checkingAccountNumber = productList.stream().filter(PackageUtility::isCheckingProduct)
                                .collect(Collectors.toMap(ProductDTO::getBankingPackageNumber,
ProductDTO::getAccountNumber));

                List<String> cins = applicantList.stream().filter(e -> (e.getCin() != null &&
!e.getCin().equals(""))).map(ApplicantDTO::getCin).collect(Collectors.toList());


        if(signerGroup.stream().map(SignerGroup::getGCIFCustomerNumber).allMatch(e -> isSignerPresent(e,
customerInformationList)))

        private boolean isSignerPresent(String gcifId, List<GetCustomerDetailInformation>
customerInformationList)
        {
                return customerInformationList.stream().anyMatch(e ->
gcifId.equals(e.getGCIFCustomerNumber()));
        }
```



```
private List<GetCustomerDetailInformation> getSignerInformationOfProduct(List<CinDetailDTO> cinDetails,
Map<String, GetCustomerDetailInformation> customerDetailMap)
        {
                return
cinDetails.stream().map(CinDetailDTO::getCin).map(customerDetailMap::get).collect(Collectors.toList());
        }
filteredList = filteredList.stream().filter(s -> null != s.getDomicileBranchState()
        && s.getDomicileBranchState().equalsIgnoreCase(governingState))
                                        .collect(Collectors.toList());
public List<AccountCinList> getAccNumberForFilenet(List<FilenetDocument> imageCaptureList) {

                List<AccountCinList> accountCinList = new ArrayList<>();

                imageCaptureList.stream().forEach(e -> {
                        AccountCinList accountCin = new AccountCinList();
                        if (e.getAccountNumber() != null && !e.getAccountNumber().isEmpty() &&
e.getAccountNumber().contains(",")) {
                                accountCin.setAccountNumber(e.getAccountNumber().split(",")[0].trim());
                        } else {
                                accountCin.setAccountNumber(e.getAccountNumber());
                        }
                        accountCin.setCin(e.getCin());
                        accountCinList.add(accountCin);
                });

                return accountCinList;

        }
```