```
// Validation rules for input fields
export const FIELD_VALIDATION_PATTERNS = {
  FIRST_NAME: {
    /**
     * Pattern accepts all alphabetical characters with accents,
     *  including umlauts
     *
     * Utilizing the "range" operator "exploits the ordering
     *  of characters in the charset to define a continuous
     *  range" (comment by @Angad, see link below)
     *
     * Adapted from https://stackoverflow.com/a/26900132/4233945
     */
    legalChar: {
      key: "legalChar",
      pattern: /^[A-Za-zÀ-ÖØ-öø-ÿ \-]{1,100}$/
    },
    illegalChar: {
      key: "illegalChar",
      pattern: /^[^?!@#$%^&*()=+:;<>,/{}[\]_0-9]{1,100}$/
    },
    minMax: {
      key: "minMax",
      pattern: /^.{1,100}$/i
    }
  },
  MIDDLE_NAME: {
    legalChar: {
      key: "legalChar",
      pattern: /^([A-Za-zÀ-ÖØ-öø-ÿ][\-]?){1,50}$/
    },
    illegalChar: {
      key: "illegalChar",
      pattern: /^[^?!@#$%^&*()=+:;<>,/{}[\]_0-9]{1,50}$/
    },
    minMax: {
      key: "minMax",
      pattern: /^.{1,50}$/
    }
  },
  LAST_NAME: {
    legalChar: {
      key: "legalChar",
      pattern: /^[A-Za-zÀ-ÖØ-öø-ÿ \-]{2,200}$/
    },
    illegalChar: {
      key: "illegalChar",
      pattern: /^[^?!@#$%^&*()=+:;<>,/{}[\]_0-9]{2,200}$/
    },
    minMax: {
      key: "minMax",
      pattern: /^.{2,200}$/i
    }
  },
  PHONE: {
    format: {
      key: "format",
      pattern: /^(1\-)?[0-9]{3}(\-)?[0-9]{3}(\-)?[0-9]{4}$/
    },
    minMax: {
      key: "minMax",
      pattern: /^.{10,40}$/
    }
  },
  EMAIL: {
    format: {
      key: "format",
      pattern: /^.+@.+\.[a-z]{2,}$/
    },
    hasAtSymbol: {
      key: "hasAtSymbol",
      pattern: /@/
    },
    minMax: {
      key: "minMax",
      pattern: /^.{6,256}$/
    }
  },
  USERNAME: {
    format: {
      key: "format",
      pattern: /^[a-zA-Z0-9\-.\_]{6,256}$/
    },
    minMax: {
      key: "minMax",
      pattern: /^.{6,256}/
    }
  },
  PASSWORD: {
    capitalAlpha: { key: "capitalAlpha", pattern: /[A-Z]+/ },
    format: { key: "format", pattern: /^[a-zA-Z0-9!#$%\-_=+<>.]+$/ },
    lowercaseAlpha: { key: "lowercaseAlpha", pattern: /[a-z]+/ },
    minMax: { key: "minMax", pattern: /^.{8,64}$/ },
    number: { key: "number", pattern: /[0-9]+/ },
    specialCharacter: {
      key: "specialCharacter",
      pattern: /[!#$%\-_=+<>.]+/
    }
  },
  STREET_ADDR_1: {
    // Uses positive lookaheads to match alphas/numerics/space characters
apositionally
    format: {
      key: "format",
      pattern: /^((?=[^?!@$%^&*()=+:;<>,{}[\]_"]*[0-9])(?=[^?!@$%^&*()=+:;<>,{}
[\]_"]* )(?=[^?!@$%^&*()=+:;<>,{}[\]_"]*[A-Za-zÀ-ÖØ-öø-ÿ])[^?!@$%^&*()=+:;<>,
{}[\]_"]*){3,300}$/i
    },
    illegalChar: {
      key: "illegalChar",
      pattern: /^[^?!@$%^&*()=+:;<>,{}[\]_"]{3,300}$/
    },
    minMax: {
      key: "minMax",
      pattern: /^.{3,300}$/
    }
  },
  STREET_ADDR_2: {
    format: {
      key: "format",
      pattern: /^((?=[^?!@$%^&*()=+:;<>,{}[\]_"]* )(?=[^?!@$%^&*()=+:;<>,{}
[\]_"]*[A-Za-zÀ-ÖØ-öø-ÿ])[^?!@$%^&*()=+:;<>,{}[\]_"]*){3,50}$/
    },
    illegalChar: {
      key: "illegalChar",
      pattern: /^[^?!@$%^&*()=+:;<>,{}[\]_"]{3,50}$/
    },
  minMax: {
    key: "minMax",
    pattern: /^.{3,50}$/
  }
},
CITY: {
  legalChar: {
    key: "legalChar",
    pattern: /^[A-Za-zÀ-ÖØ-öø-ÿ \-\.']{2,100}$/
  },
  illegalChar: {
    key: "illegalChar",
    pattern: /^[^?!@#$%^&*()=+:;<>,{}[\]_"]{2,100}$/
  },
  minMax: {
    key: "minMax",
    pattern: /^.{2,100}$/
  }
},
STATE: {
  format: {
    key: "format",
    pattern: /^[a-z]{2}$/i
  }
},
POSTAL_CODE: {
  format: {
    key: "format",
    pattern: /^[0-9]{5}$/
  },
  minMax: {
    key: "minMax",
    pattern: /^.{5}$/
  }
},
COUNTRY: {
  minMax: {
    key: "minMax",
    pattern: /[a-z]{3}/i
  }
},
PLEDGE_ID: {
  format: {
    key: "format",
    pattern: /^([0-9]+(,?[ ]*)?\n?)+$/
  }
},
CURRENCY: {
  format: {
    key: "format",
    pattern: /^[0-9]+(\.[0-9]{1,2})?$/
  }
},
GIFT_AMOUNT: {
  min: {
    key: "min",
    func: inputValue => parseInt(inputValue) >= 5
  }
},
CC_NUMBER: {
  format: {
    key: "format",
    pattern: /^[0-9]{8,16}$/
  }
},
CC_NAME: {
  format: {
    key: "format",
    pattern: /^.+(\s.+\.?)?\s.+$/
  }
},
CC_EXPIRATION: {
  format: {
    key: "format",
    pattern: /^[0-9]{2}$/
  },
  validMonth: {
    key: "validMonth",
    func: inputValue => {
      const valueLength = inputValue.length;
      const monthDigit = parseInt(inputValue);

      /**
       * Because the month input field accepts a leading zero, check
       *  value's length before checking if the digit is zero
       */
```

```
        const isNotZero = valueLength > 1 ? monthDigit > 0 : true;

        return isNotZero && monthDigit <= 12;
      }
    }
  },
  CC_CVV: {
    format: {
      key: "format",
      pattern: /^[0-9]{3,4}$/
    }
  },
  CANCEL_REASON: {
    hasValue: {
      key: "hasValue",
      pattern: /^[a-zA-Z ]+$/
    }
  },
  CANCEL_COMMENT: {
    minMax: {
      key: "minMax",
      pattern: /^.{0,500}$/
      // func: inputValue => inputValue.length <= 500
    }
  }
};

// Static resources associated with validation patterns
export const FIELD_VALIDATION_RESOURCES = {
  PASSWORD: {
    /**
     * Ignore strings for RegExp constructor consumption from being formatted
against
     *  normal string standards
     */

    // prettier-ignore
    specialCharacters: "!#$%\\-_=+<>."
  }
};
```