CSCI P445
Software Detailed Design

Cheyenne Pierpont
Bailey Forbes
Brandon Maurer

Reference: https://en.wikipedia.org/wiki/Software_design_description
The SDD should contain the following information:

1. The *data design* describes structures that reside within the software. Attributes and relationships between data objects dictate the choice of data structures.
    - Three system applications will be used to keep this app running with the database. The first application we have is Flask Python Framework: Flask is a small and lightweight Python web framework that provides useful tools and features that make creating web applications in Python easier. The team chose to use a flask because it is easy to use and it is very flexible, making it a good choice for a new project like RiverWay. Flask can pair with different libraries so it helps give some flexibility again on which library is in use and withholding a lot of information in the libraries. Flask also can handle a lot of information and traffic all at once. In June and July, Cave Country Canoe can see up to 400 or more guests in a day. So the application has to be able to handle a lot of traffic at once.
    - The next system that will be used in the system is MySql, which is an open-source relational database management system. It is based on structured query language. MySQL is used by a lot of different developers. This application is also very reliable and is needed for any time when the paddlers are on the water these coming years. MySQL is very easy to use and it is a high-performing application. It is like flask It is very flexible but also has a lot of security features. The security features are another aspect that Cave Country Canoes needs; they do not want just anyone to be able to ask about the locations of their customers. They will only want certain employees to see these locations and respond in emergencies.
    - Heroku is our last application software inside the app. It is a cloud - based platform that helps developers build, run, and manage applications. Heroku handles the infrastructure so that Cave Country Canoes can focus on making sure the app is up and running and helping with emergencies. Heroku is very flexible and you can use many different languages inside of it. If the developers or owners need to make changes to the application or add new features it handles it smoothly and easily. The service is low cost and it pairs with other add-on providers with the low prices. If you are upgrading a system and need to release an older release it will roll back and host that while you fix and upgrade the new version as well.
2. The *architecture design* uses information-flowing characteristics and maps them into the program structure. The transformation mapping method is applied to exhibit distinct boundaries between incoming and outgoing data. The data flow diagrams allocate control input, processing, and output along three separate modules.
    - note reference Riverway 2024-2025 database pdf for explanation.
    - note reference Database tables on SAS RF-3
3. The *interface design* describes internal and external program interfaces, as well as the design of the human interface. Internal and external interface designs are based on the information obtained from the analysis model.

- The interface design of our program begins with an app icon on the screen of a smartphone. When the user clicks on the icon the app will open up to a login screen. There will be a prompt instructing the user to log in if they already have an account or to sign up if they do not. If the user chooses to sign up the screen will change to the signup screen which has more boxes for the user to fill out. If they choose to log in and their information is not accepted a message will appear letting the user know. If they log in and their information is accepted the screen will change to bring them to the rest of the app.
    - Currently, this design is still being worked on and tweaked, but as of the moment the screen changes to show the map of the river and at the bottom of the screen a button that says send location. The user is then able to tap on the map which will place a red dot to indicate their location and then hit the button to send it. We also want to include a section that says a little history about the customer. We want to include an area for app settings as well. That is the design of the interface as we see it now, but is very likely to see future changes and edits.
- The cipher we will be providing will be through a CLI-like interface. A CLI allows for modularity for independent use, and CLIs can easily be used through executables or the Python command - the cipher is made in Python.

4. The ***procedural design*** describes structured programming concepts using graphical, tabular, and textual notations. These design mediums enable the designer to represent procedural detail that facilitates translation to code. This blueprint for implementation forms the basis for all subsequent software engineering work.

Customer Procedures

1. customer account creation and login
    - purpose: allow customers to create an account, log in, and access the customer portal.
    - steps:
        - customer provides account details including username, password, email, phone number, address, emergency contact information, and other necessary details.
        - system stores the account in the main Account table and links customer-specific data to the Customer table.
        - system assigns default permissions based on the customer role.
        - once created, the customer can log in through the customer portal.

          pseudo code:

```
- function createCustomerAccount(details):
  accountID = insertIntoAccount(details)
  insertIntoCustomer(accountID, details.emergencyContact)
  setPermission(accountID, 'Customer')
  return accountID
```

```
function loginCustomer(username, password):
if validateCredentials(username, password):
displayCustomerPortal()
else:
    showError("Invalid login")
```

2. logging into the system and accessing the customer portal
    - purpose: allow customers to log into the system, leading them to the customer portal.
    - steps:
        - customer enters username and password.
        - system verifies credentials against the Account table.
        - Upon successful login, the customer is directed to the customer portal where they can access options like check-in, emergency ping, points of interest, and leaving a review.
        - The navigation bar at the top displays options like log out, send emergency ping, check points of interest, and leave a review.

          pseudo code:

```
- function customerLogin(username, password):
  if verifyCredentials(username, password):
      navigateTo(customerPortal)
      displayNavigationBar(["Log out", "Send Emergency Ping", "Check Points of Interest", "Leave Review"])
  else:
      showError("Login failed. Please check credentials.")
```

3. creating an emergency ping
    - purpose: allow customers to send an emergency alert in case of distress.
    - steps:
        - customer selects the emergency ping option.

          system prompts the customer to confirm the emergency ping.

          system records the emergency ping in the EmergencyDistressAlerts table with the customer’s location and timestamp.
    - system queues the alert for review by the emergency team.

      pseudo code:

```
- function createEmergencyPing(customerID, location):
  confirm = promptUser("Confirm emergency ping?")
  if confirm:
      alertID = insertEmergencyAlert(customerID, location)
      queueForReview(alertID)
      showMessage("Emergency ping sent successfully.")
  else:
      showMessage("Emergency ping canceled.")
```

4. checking in for a trip

- purpose: allows customers to check in for their trip, marking the start of their experience.
- steps:
  - customer selects the check-in option on the main menu.
  - system updates the CustomerTrip table to reflect that the customer has checked in.
  - check-in confirmation is displayed to the customer, and they are redirected to the main menu.
  - pseudo code:

```
- function checkInForTrip(customerID, tripID):
  if tripExists(tripID) and notAlreadyCheckedIn(customerID, tripID):
      updateCustomerTripStatus(customerID, tripID, "Checked In")
      showMessage("Check-in successful.")
      redirectToMainMenu()
  else:
      showError("Unable to check in. Please verify your trip details.")
```

5. viewing points of interest
   - purpose: display locations of interest to customers, enhancing their experience.
   - steps:
     - customer selects the points of interest option.
     - system retrieves locations from the PointsOfInterest table and displays them.
     - customer can view details about each point of interest.

     pseudo code:

```
- function viewPointsOfInterest():
  points = fetchAllPointsOfInterest()
  displayPoints(points)
```

6. leaving a review on trip or service

- purpose: allow customers to leave feedback on their trip or experience.
- steps:
  - customer selects the leave a review option.
  - system prompts for the trip ID and rating or comments.
  - system saves the review in the CustomerFeedback table.
  - The customer is notified that their feedback was successfully submitted.

  pseudo code:

```
- function leaveReview(customerID, tripID, rating, comments):
  if isValidTrip(tripID) and isValidRating(rating):
      feedbackID = insertFeedback(customerID, tripID, rating, comments)
      showMessage("Thank you for your feedback!")
```

```
else:
    showError("Invalid review details.")
```

7. logging out
    - purpose: allow customers to log out from their session securely.
      steps:
      customer selects the log out option from the navigation bar.
      system terminates the customer session and redirects them to the login screen.
        - pseudo code:

```
- function customerLogout():
  terminateSession()
  redirectToLogin()
```

Employee Procedures

1. check customers in distress
    - purpose: allow employees to view all active distress alerts.
    - steps:
        - employee selects the distress alerts option.
        - system retrieves all unresolved distress alerts from the EmergencyDistressAlerts table.
        - distress alerts are displayed to the employee, including location and severity details.
            - pseudo code:

```
- function viewDistressAlerts():
  alerts = fetchUnresolvedDistressAlerts()
  displayAlerts(alerts)
```

- confirm customer safety
  purpose: allow employees to mark customers as safe, resolving their distress status.
  steps:
  employee selects an active distress alert.
  system updates the distress alert status to resolved.
  system removes the assignment from the employee and updates the CustomerTrip status.
    - pseudo code:

```
function confirmCustomerSafe(alertID, employeeID):
if isAssignedToEmployee(alertID, employeeID):
    updateDistressAlertStatus(alertID, "Resolved")
    unassignEmployeeFromAlert(alertID, employeeID)
    showMessage("Customer marked as safe.")
```

```
        else:
            showError("You are not assigned to this alert.")
```

- logging into the employee portal
  purpose: allow employees to log in and access their dashboard.
  steps:
  employee enters username and password.
  system verifies credentials and grants access to the employee portal.
    - pseudo code:

```
function employeeLogin(username, password):
    if verifyCredentials(username, password, "Employee"):
        navigateTo(employeePortal)
    else:
        showError("Login failed.")
```

4. view routes, trips, and geofences
   purpose: provide employees with visibility into available routes, trips, and geofences.
   steps:
   employee selects the view routes or view trips option.
   system retrieves relevant data from the Route, Trip, and Geofence tables.
   employees can view route details, trip schedules, and geofence boundaries.
    - pseudo code:

```
function viewRoutesTripsGeofences():
    routes = fetchAllRoutes()
    trips = fetchAllTrips()
    geofences = fetchAllGeofences()
    displayData(routes, trips, geofences)
```

Admin Procedures

1. create employee account
   purpose: allow admins to create employee accounts.
   steps:
   admin enters employee details including username, position, and permissions.
   system creates a new account in the Account table and links it to the Employee table.
    - pseudo code:

```
- function createEmployeeAccount(details):
  accountID = insertIntoAccount(details)
  insertIntoEmployee(accountID, details.position)
  setPermission(accountID, 'Employee')
  return accountID
```

2. create routes, points of interest, and geofences
    - purpose: allow admins to set up new routes, points of interest, and geofence boundaries.
      steps:
      admin selects the option to create route, POI, or geofence.
      system prompts for necessary details based on selection.
      system saves the information in the respective table.
        - pseudo code:

```
- function createRoute(details):
  routeID = insertIntoRoute(details)
  showMessage("Route created successfully.")

  function createPOI(details):
  poiID = insertIntoPointsOfInterest(details)
  showMessage("Point of Interest created successfully.")

  function createGeofence(details):
  geofenceID = insertIntoGeofences(details)
  showMessage("Geofence created successfully.")
```

Super Admin Procedures

1. create admin account
    - purpose: allow super admins to create admin accounts.
      steps:
      super admin enters account details for a new admin.
      system stores the account in the Account table and assigns appropriate permissions.
        - pseudo code:

```
function createAdminAccount(details):
accountID = insertIntoAccount(details)
setPermission(accountID, 'Admin')
return accountID
```

2. full key access in shred cipher
   - purpose: allow super admins to access and manage the full cryptographic key.
     steps:
     super admin accesses the shred cipher management interface.
     system grants super admin access to full key and state files for encryption and decryption operations.
     - pseudo code:

```
- function accessShredCipherFullKey():
  if userRole == 'Super Admin':
      provideFullKeyAccess()
  else:
      showError("Insufficient permissions.")
```

These procedures are designed to handle the workflows and responsibilities of each user role in the system, including data interactions and access control.

5. Breakdown of individual contributions

   - Cheyenne Pierpont - Assisted in filling out RF-4, Contacting the customer for information, and keeping her in the loop. Helping in front-end research and development. Installed MySQL Workbench and added in new connections with getting the CCC_Emergency_Map on MySQL Work Bench.

   - Bailey: Worked on the back end of the system and helped out-group in setting up MySQL Workbench local sessions.

   - Brandon Maurer - Assisted in filling out RF4, continued working on the Front End of project, installed MySQL Workbench and learned how it works

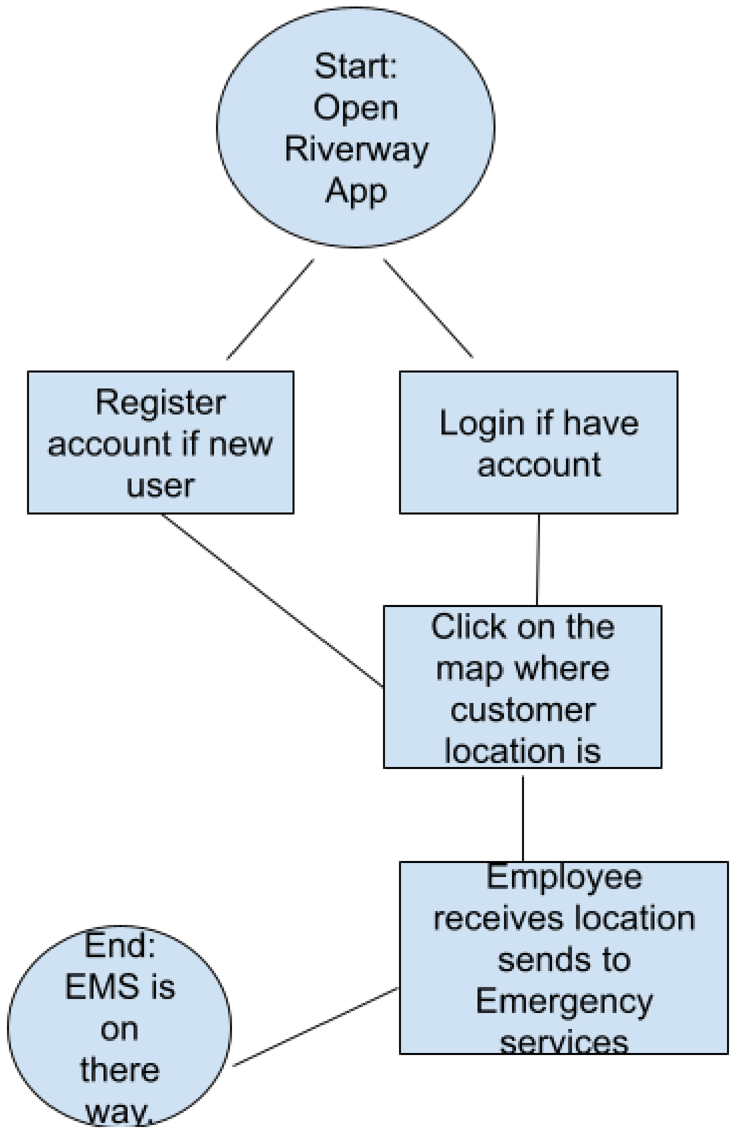
6. Key Personnel information
   - Cheyenne Pierpont - cpierpon@iu.edu or cheyennepierpont@outlook.com
   - Bailey Forbes - bailforb@iu.edu or prestonforbesbpf@gmail.com
   - Brandon Maurer - bramaur@iu.edu or brandondmaurer@gmail.com

Each procedural component should include one section devoted to the detailed description of the component
- Introduction/Purpose of this Component/Entity

    - Our project will be designed to help Cave Country Canoes improve the management of emergency distress situations and the handling of map-based data on their operations, among the critical tasks that are currently being done by them. With the emergence of such a modern system, the management of processes will be made more efficient, integrating the following:
        - Emergency incident tracking
        - Customer feedback management
        - Route and trip condition logging
        - Resource allocation for emergency response teams
- Input for this Component/Entity
    - The Input for this app will be from the customer when they are having an emergency situation on the river. The customer will come to a login screen and they will login to their account. After logging into the app they will come to the Cave Country Canoes map on the next page.On this map there will be a button saying “Contact Representative from Cave Country Canoes with location for emergency”. The user will send their location from the map to CCC. Essentially this will be sent to who is working that night and the information can be sent to the right Emergency services and get help to them faster.
- Output for this Component/Entity
    - The output will be that a representative at Cave Country Canoes will receive a customer's message with their location in an emergency situation. With this they will forward the information to the correct Emergency services and send people out in a boat to get to them as quickly and safely as they can.
- Component/Entity Process to Convert Input to Output
    - Once the customer has downloaded the app and opened it from the app store which could be from Apple App Store, Google Store, or App store, etc. They will log into the app that will be hosted on Flask and Heroku. Once they input their data for their location it will be stored in the database that will be stored and held for a certain amount of time on MySql. MySql will then send the information through the app back to the employee at the time who is responsible for these locations and for forwarding this information. This information should and will be sent to the appropriate Emergency services depending on the situation.
- Design constraints and performance requirements of this Component/Entity
    - A performance requirement for this app will be the customer having their phone handy on the river.
        - if they do not have their phone on the river they of course do not have access to the usefulness of the app.
    - Another performance requirement for the app to work as well is being able to connect to the internet or having a service connection along the river.
        - This is a performance requirement as well as a design constraint. If the paddler does not have service at the time of the emergency. They may not be able to contact Cave Country Canoes through the app. At that point they would need to call 911.
    - Another Design Constraint / Risk for the app is server migration: which will happen every 3-4 years when new hardware is procured, data migration would be handled through bulk data

export/import using tools like mysqldump for MySQL and reconfiguring the Heroku deployment.

- Process (pseudo-code algorithm)
  - o This is based on the fact that they have the app downloaded already and this is a situation when on the river.
  - o See submitted ER diagram PDF for a more in-depth database structure plan


Start: Open Riverway App
Register account if new user
Login if have account
Click on the map where customer location is
Employee receives location sends to Emergency services
End: EMS is on there way.

The product to be produced is a system specification.

You should be able to pass the document off to another team of very junior programmers and they could build that system.

All programs should be described in detail, list all major methods/routines and the English one sentence description of each method/routine. And all should be very computable (no magic occurs here).

The database (or all data store) should be described at the ER Diagram level.

Products for hardware platform, database and language for all programs should be specified.
Also, this report should be posted to the team website.

Portions of previous assignments can be used within this document in appropriate sections.

Note: Portions of this document are from https://en.wikipedia.org/wiki/Software_design_description .