SHI YANCHEN

A WEATHER APPLICATION ON INFOTAINMENT SYSTEM

SYSTEM COMPONENTS

- Front-end
  - HTML 5 client
- Local back-end
  - Database
  - Snapshots
- Remote back-end
  - Weather server

FRAMEWORK

- Django
  - Model:
    - retrieve data from local database and snapshots
    - retrieve data from remote weather server and update database
  - Template
    - HTML and CSS
  - View
    - Render template given retrieved data from model

INTERFACES

- Case 1:
  - Only current location is presented
- Case 2:
  - Both current and destination locations are presented
- Location search
  - Fuzzy search of available locations

PERMISSIONS

- Location service
- Notification
  - Alarm the driver when the back-end retrieves a change of weather on current location or destination
- Connection with navigation system
  - Set destination automatically if the navigation is running

FRONT-END: DATABASE OR SNAPSHOT

- Three categories of locations
  - current location
    - update frequently; query by longitude and latitude
  - destination location
    - update frequently; query by the location name
  - others
    - update manually; query by the location name

RESTFUL API

- Resource Representational State Transfer
  - Resource: a backend weather server supporting URL query
  - Representational: JSON
  - State Transfer: HTTP methods
    - the frontend in infotainment using only GET method
    - POST and PUT methods are used by backend

RESTFUL API: URL DESIGN

- URL root
  - specifying API, like api.jlr.com/weather
- Using norms:
  - like GET api.jrl.com/weather/shanghai
- Well designed status code

RESTFUL API

- GET: api.jlr.com/weather/locations
  - retrieve all possible locations, and stored locally for further fuzzy search
- GET: api.jlr.com/weather/shanghai
  - retrieve weather given a location
- GET: api.jlr.com/weather/@31.297344,121.5030465
  - retrieve weather given a longitude and latitude

TEXT

THANKS