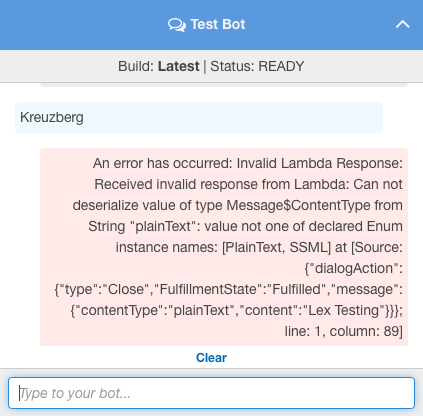
**Contents**

* Abstract, list of figures, Glossary, etc.
* Introduction
  + Motivation
    - alternative approaches to a classical information system
    - Use Cases
  + [Research Question development/Proposal/Anforderungen/…]
* Available Implementation
  + Functionality of the current Berlin.de bot
  + Data Structure (current JSON nodes)
  + Static / Dynamic Data
* Available Technologies (Case Study)
  + Amazon Lex
    - Features / Strengths / Weaknesses
  + Firebase api.ai Firebase
    - Features / Strengths / Weaknesses
  + Microsoft Azure bot Framework (beta)
    - …
  + Google Chatbase Analytics
* Design Specs
  + Technologies Used
  + Functionality
    - Static / Dynamic Data
    - Recommended Citizen Services
    - Booking an Apointment
  + Why certain decisions were met
* Implementation
  + Project Setup
  + Modules (Python /Node.js)
  + Performed unit tests
  + Scalabilty / Extension Feasibility
* Platform Integration
  + Facebook
  + Amazon (Alexa/Lex)
  + Slack
* Usability
  + Tests (Performance Evaluation)
* Conclusion
  + Social Aspect
  + Economic Value
* Bibliography



* Lex
  + Context is kept over defined session duration (-) user cannot control session duration /
  + can manage flow of a conversation
  + Utterances bereichert von Natural Language Understanding
  + Error Handling: It tells you which utterances were missed to train the system manually in addition
  + Auto Scaling
  + Fulfillment through Amazon Lambda
  + Suggestions etc can be done with Lambda
  + Identity Access Management (IAM)
  + Slots (and Slot Types) cannot be imported, have to be added manually
  + undo/redo within intent interface not possible, within code only possible with restoration (poor IDE)
  + user has to spell the slot name properly? works bad with non-native Names
* API.AI
  + Like REST, it is trying to be more of a standard to be integrated into many applications
  + Sends fulfillment information via webhook
* Developer’s role
  + clarity
  + spontaneity
  + competence