iFarm - advertisement platform for local producers

Motivation

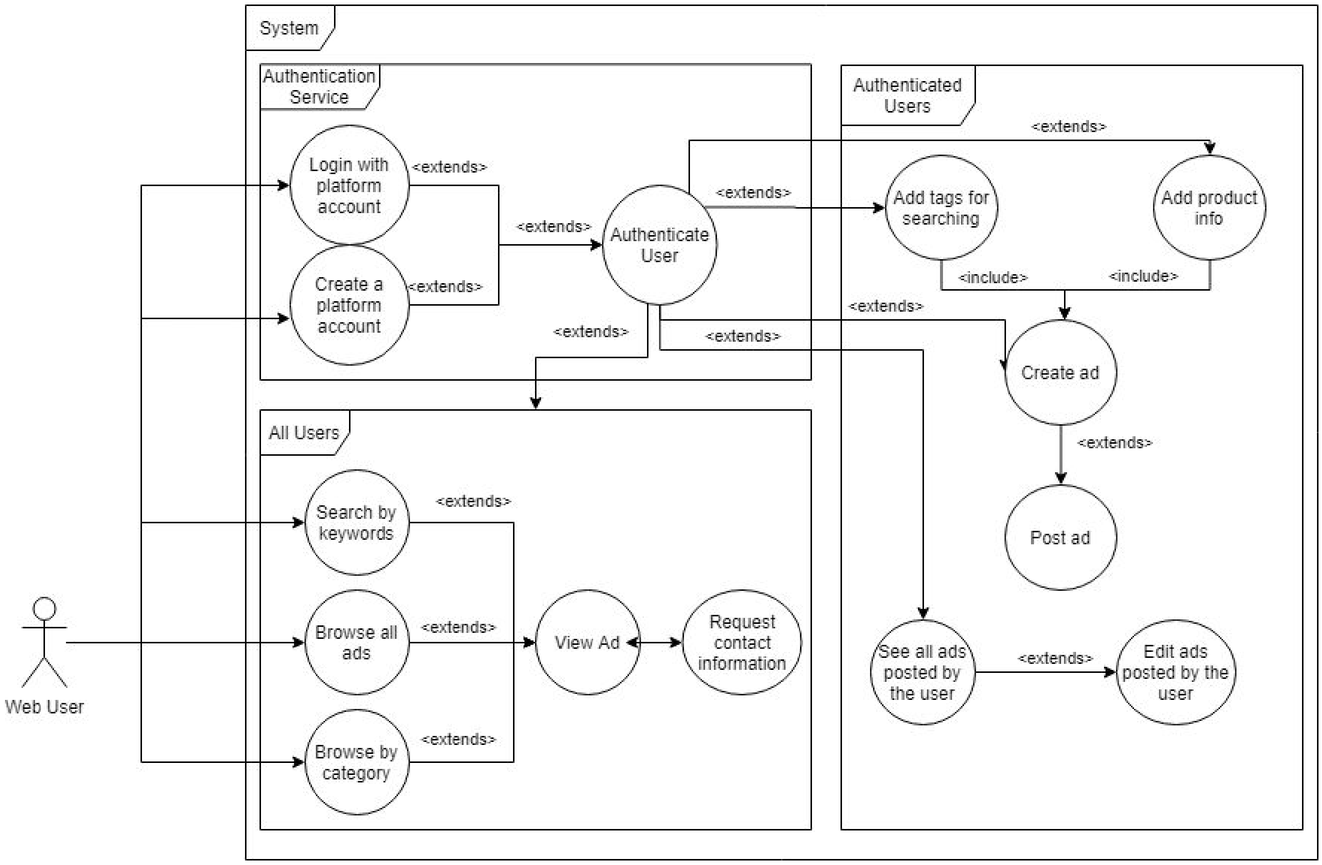
The recent situation has provoked a grave problem for the national and international community, having most of the industry come to a standstill thanks to novel coronavirus, COVID-19. As such, our government is trying to reduce the impact of this crisis and rushing to get our economy to the way it was before. However, the situation before this pandemic is not an ideal one, having most of our products come from abroad, wasting money for goods that could be procured from local producers. From an agricultural side, our country has a lot of arable land which can yield a lot of produce, however, it is not our main source of product. From an industrial side, we have proven in the communism era that we have much industrial power, but it was all shut down after the regime change. And from a natural resources standpoint, our country has gas, oil, gold and silver among others, however we seem to be exporting them at insulting prices. As such, we believe that, to start making the most out of our potential, a good first step is to make the population aware of our local producers and manufacturers, and to create a platform where they can advertise their product and find others from whom they can procure goods that are necessary to maintain their production.

Techstack

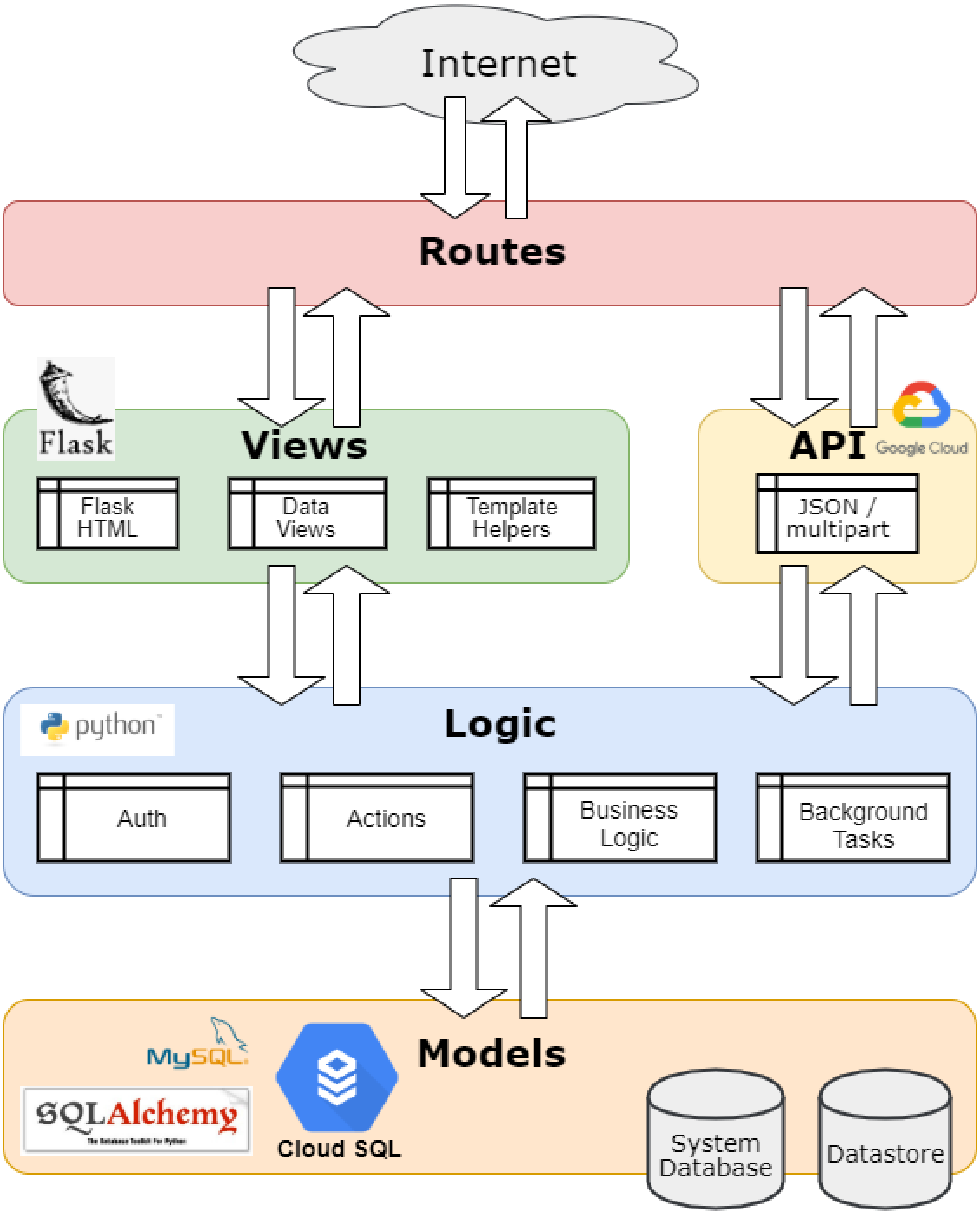
Techstack:

1) Front-end:
   - HTML, CSS, Javascript
   - Jquery
2) Back-end:
   - Python - server-side language
   - Flask - web framework
   - MySQL - database engine
   - SQLAlchemy - for database access in the application
3) Cloud APIs and services:
   - Logging and error reporting - for monitoring the health and flow of the application
   - Cloud Storage - database hosting
   - Google Maps API and Geolocation Api - for giving a precise location of the advertiser
   - Debugging - to detect and solve errors in our application in a fast and efficient manner
   - Cloud Translation - multi-language support
   - Text-to-Speech - accessibility; for users that have disabilities which can make using our platform difficult (dyslexia)
   - ReCaptcha API - for security against attacks using automated software
   - OpenApi - not to be confused with the OpenAPI format, although not a Cloud service, it is necessary for making sure that only local companies and independent producers can use our platform, as that is who we wish to support

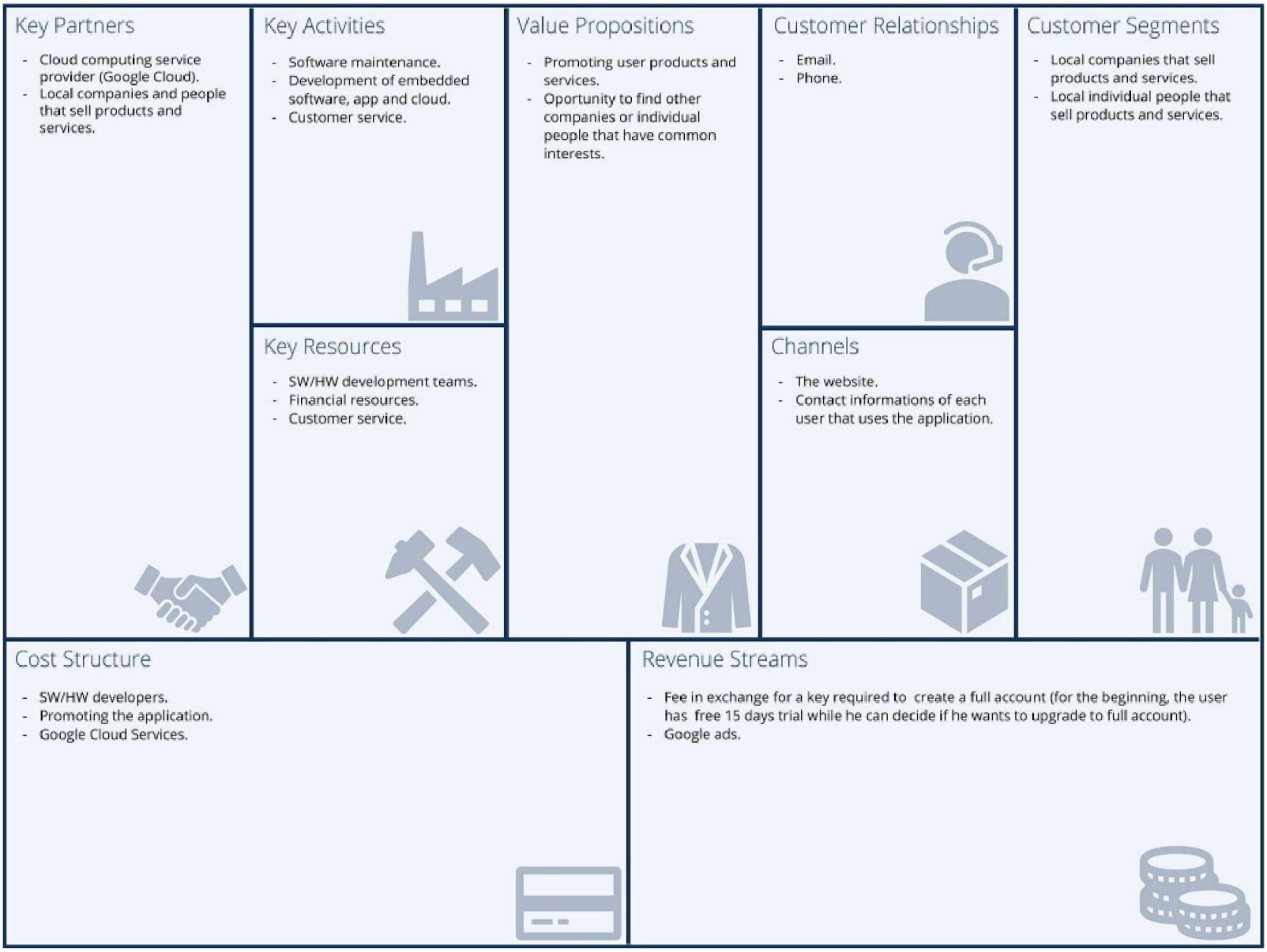
Use Case diagram



Architectural Diagram



Business Model Canvas



Key Partners

- Cloud computing service provider (Google Cloud).
- Local companies and people that sell products and services.

Key Activities

- Software maintenance.
- Development of embedded software, app and cloud.
- Customer service.

Key Resources

- SW/HW development teams.
- Financial resources.
- Customer service.

Value Propositions

- Promoting user products and services.
- Oportunity to find other companies or individual people that have common interests.

Customer Relationships

- Email.
- Phone.

Channels

- The website.
- Contact informations of each user that uses the application.

Customer Segments

- Local companies that sell products and services.
- Local individual people that sell products and services.

Cost Structure

- SW/HW developers.
- Promoting the application.
- Google Cloud Services.

Revenue Streams

- Fee in exchange for a key required to create a full account (for the beginning, the user has free 15 days trial while he can decide if he wants to upgrade to full account).
- Google ads.

Api

```
openapi: "3.0.0"
info:
  version: 1.0.0
  title: <title>
  description: A API that lets the users of our platfrom interogate the advertisment ads and modify theyr visibility
  termsOfService: <API>
  contact:
    name: <name>
    email: <email>
    url: <URL>
  license:
    name: Apache 2.0
    url: https://www.apache.org/licenses/LICENSE-2.0.html
servers:
  - url: <URL>/api
```

```yaml
paths:
  /ads/{ID}:
    get:
      description: Gets all the ads that the user has posted based on ID.
      operationId: Find ads by user ID
      parameters:
        - name: id
          in: path
          description: ID of user to fetch ads from
          required: true
          schema:
            type: string
            items:
              type: string
      responses:
        '200':
          description: Ads response
          content:
            application/json:
              schema:
                type: array
                items:
                  $ref: '#/components/schemas/ADS'
        default:
          description: unexpected error
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Error'
```

```
/self_ads:
  get:
    description: Gets all the ads that the user has posted.
    operationId: Find ads of the user
    parameters:
      - name: api_key
        in: query
        description: Api key of the user
        required: True
        schema:
          type: String
          format: String
    responses:
      '200':
        description: Ads of the User
        content:
          application/json:
            schema:
              type: array
              items:
                $ref: '#/components/schemas/ADS'
      default:
        description: unexpected error
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Error'
```

```
/post_Ad:
  post:
    description: Creates a add for the user:
    operation: Inserts ad in the db:
    parameters:
      - name: api_key
        in: querry
        description: api key of the user
        required: True
        style: form
        schema:
          type: string
          items:
            type: string
      - name: description
        in: querry
        description: description of the posting
        required: True
        style: form
        schema:
          type: string
          items:
            type: string
      - name: title
        in: querry
        description: title of the posting
        required: True
        style: form
        schema:
          type: string
          items:
            type: string
    responses:
      '200':
        description: add has been posted succefully
        content:
          application/json:
            schema:
              type: array
              items:
                $ref: '#/components/schemas/succesful_post'
```

```yaml
default:
  description: unexpected error
  content:
    application/json:
      schema:
        $ref: '#/components/schemas/Error'
```

```
/disable_ad:
  put:
    decription: Activates or deactivates a certain ad
    operation: Modifies "active" filed in db
    parameters:
      - name: api_key
        in: querry
        description: api key of the user
        required: True
        style: form
        schema:
          type: string
          items:
            type: string
      - name: posting_id
        in: querry
        description: id of the posting
        required: True
        style: form
        schema:
          type: string
          items:
            type: string
      - name: status
        in: querry
        description: status value(true or false)
        required: True
        style: from
        schema:
          type: boolean
```