Table of Contents

1. Task

To create application useful for users to control their to-do list. Each user can access and modify his own to-do list. Application should be created using new technologies. Application type – web based.

1.1. Application usability

Application will be used for big group of people simultaneously. First application will be used as browser based, but in future will be developed mobile application. For this reason, system must provide data for mobile applications.

1.1.1. Users

User will use system to control to-do list. User actions:

- Register using personal data;
- Login to application;
- Add new to-do;
- Confirm that to-do is done;
- Remove to-do;

1.1.2. To-do list

To-do list should be represented clearly. To-do list should contain information:

- To-do information (description text);
- Creation date;
- Completion date;
- To-do is completed or not;

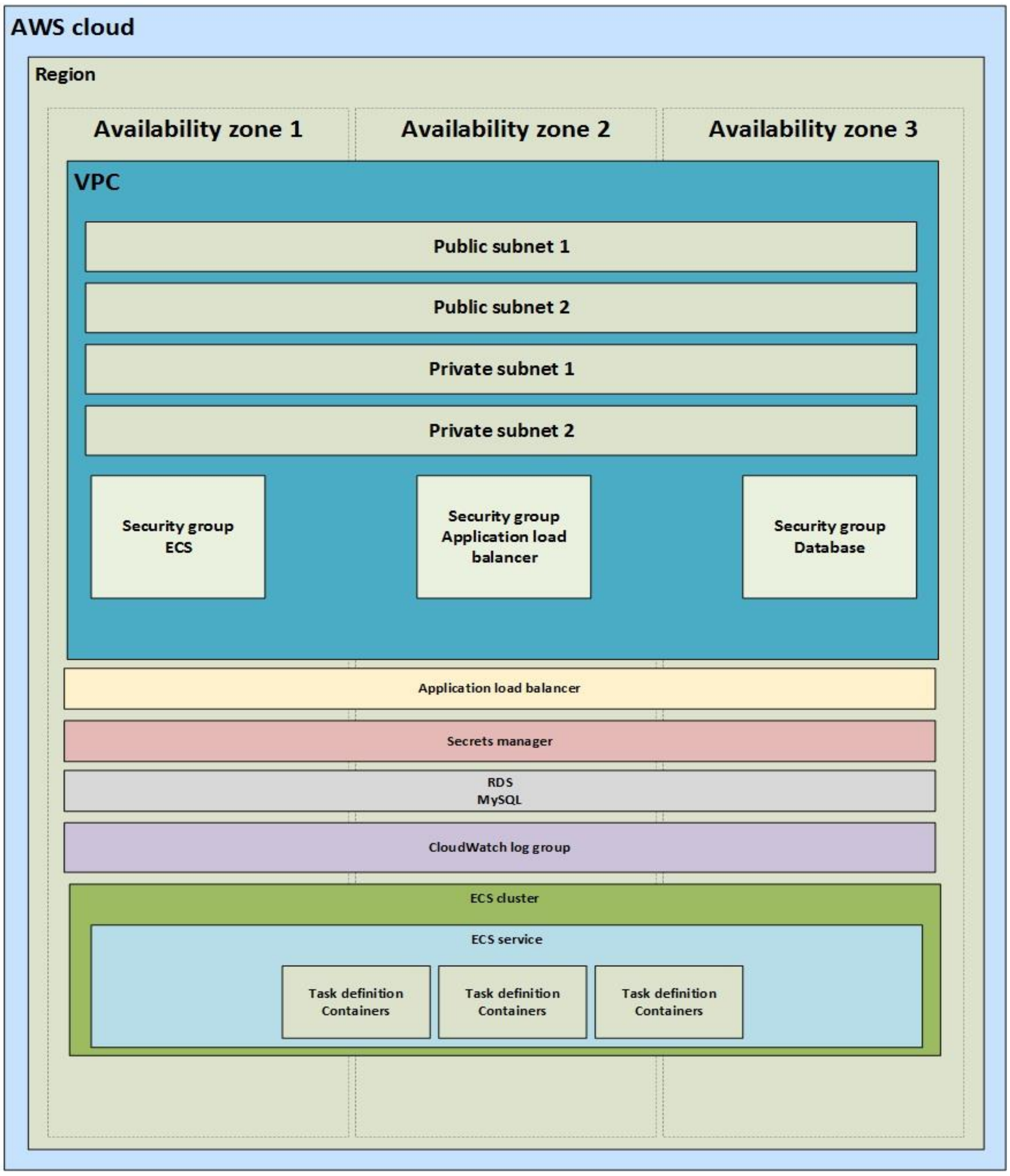
1.2. Technical requirements

Application must be deployable in could environment. Brief list of requirements:

- Deployable in cloud (AWS) environment;
- Infrastructure written in code;
- RDS MySQL database. Database disabled form access from outside;
- Application must be containerized;
- Application must be cost effective;
- Application must be written in Python programming language;
- Application must use Flask framework;
- Application and infrastructure code must have different repositories.
- Code must be versioned using version control system git;

2. Infrastructure

As required, infrastructure will be deployed to AWS environment. Infrastructure will be realized as IaC (infrastructure as a code) using Terraform and Terraform AWS provider library. For code versioning will be used git version control system and github.com repository. Using the given requirements, the architecture of the resources needed to complete the task was created:



According this block diagram resources will be created.

2.1. Virtual private cloud

For virtual cloud (VPC) creation was made a module named: my_vpc. This module automatically creates:

- VPC itself;
- Public subnets;
- Private subnets;
- Assigns subnet to availability zone;
- Internet gateway;
- Route table;
- Subnets (private) group for database;

Module variables:

Variable name	Type	Definition
Inputs		
vpc_name	string	Name of VPC.
vpc_cidr_range	string	CIDR block for VPC.
public_subnets	list(string)	List of subnets to create inside VPC with internet accessibility.
private_subnets	list(string)	List of subnets to create inside VPC without internet accessibility.
azs	list(string)	List of availability zones to be used.
Outputs		
public_ids	-	Public subnet-(s) id-(s).
private_ids	-	Private subnet-(s) id-(s).
vpc_id	-	VPC id.
db_subnet_gr_name	-	Subnets group name. Will be used for database creation.

When using my_vpc module, you need to assign input variables (mentioned above). Module outputs variables for usage in other configuration parts.

Public subnets used for resources with accessibility from outside the cloud. Private subnets used to restrict accessibility from outside.

2.2. Security group

Security group works like firewall. Security group blocks incoming and outgoing traffic. For this project to complete, needs three security groups:

- Security group for application load balancer (ALB);
- Security group for elastic container service (ECS);
- Security group for relational database service (RDS);

Created module for security group. Module variables:

Variable name	Type	Definition
Inputs		
vpc_id	string	VPC id in which security group works.
environment_name	string	Environment name to tag resource.
Outputs		
alb_sq_id	-	Application load balancer security group id. Used in load balancer.
ecs_sg_id	-	Elastic container service security group id. Used in ECS.
db_sg_id	-	Database security group. Used for database.

2.3. Logs

Using native CloudWatch for logging. CloudWatch will log containers stdout. Creating log group to combine logs and log stream. In task definition file we only register log group.

2.4. Secrets manager

Secrets manager is used to protect sensitive data. Secrets managers ensures safe sensitive data provision to resources. Secrets, for this project, will be created manually by AWS cloud administrator:

- **db_creds** – this secrete used to connect to database. Secret properties:
    - db_username;
    - db_password;
    - db_host;
    - db_name;
- **registry_creds** – this secret used to connect to docker hub registry. Secret properties:
    - username;
    - password;

2.5. IAM policies

Changed “ecsTaskExecutionRole” role policies. Appended two policies:

- Read secrets from secrets manager;
- Log stdout to CloudWatch log group.

2.6. Database

Using AWS RDS. Selected database – MySQL. Database uses private subnets group to restrict accessibility from outside. Secrets for database is used from secrets manager. Security group defined by ***db_sq_id*** [1].

2.7. Application load balancer

Application load balancer used to ensure ECS tasks stability and reduce response time. Application load balancer divides incoming traffic to ECS tasks. This way the response time is as minimum as possible.

Load balancer uses security group and public subnets.

Load balancer has target group and listener. Listener are responsible to catch configured traffic and redirect it to target group. Target group using configuration redirects traffic to targets.

2.8. Elastic container service

Elastic container service ensures that tasks (container) will be created and started. ECS monitors task parameters and if needed takes actions to ensure stability. ECS can reload tasks if current state is “unhealthy”, can auto-scale to increase or decrease tasks number. This service decided to use, because it takes a lot of work for us and is cost efficient (pay-as-you-go).

To activate ECS, resources was created:

- Cluster – cluster used for services. One cluster could have more than one service;
- Service – service lives in cluster. Service is used to serve container. In service configuration is set network configuration, assigned load balancer. It is like sand box for tasks. Service could have one or more than one task.
- Task definition – task definitions define the task capabilities (cpu, memory, network mode). In task definition also configures container image properties.

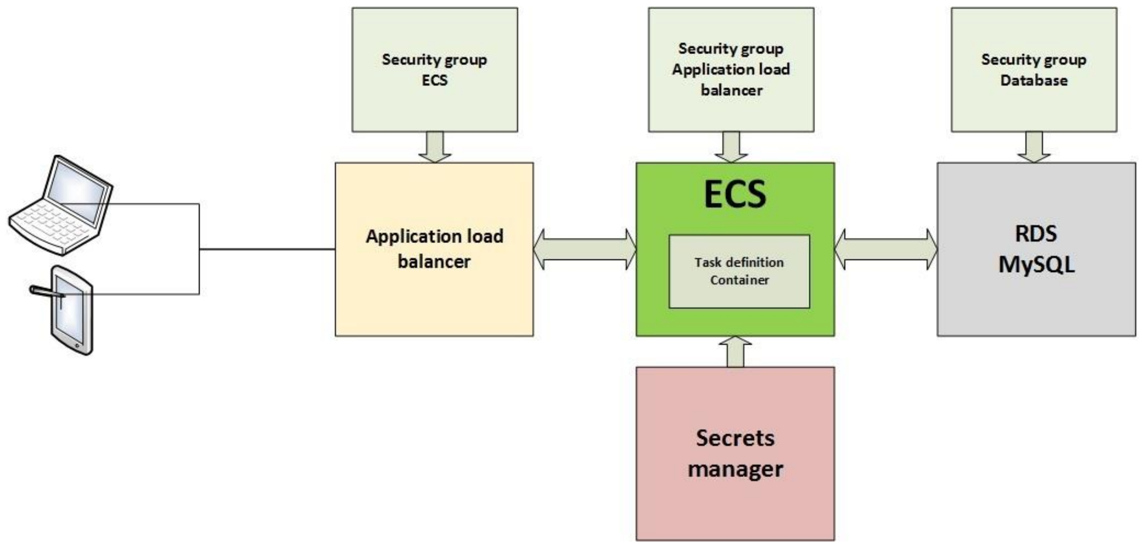
[1] 2.2. Security group

2.9. Conclusion

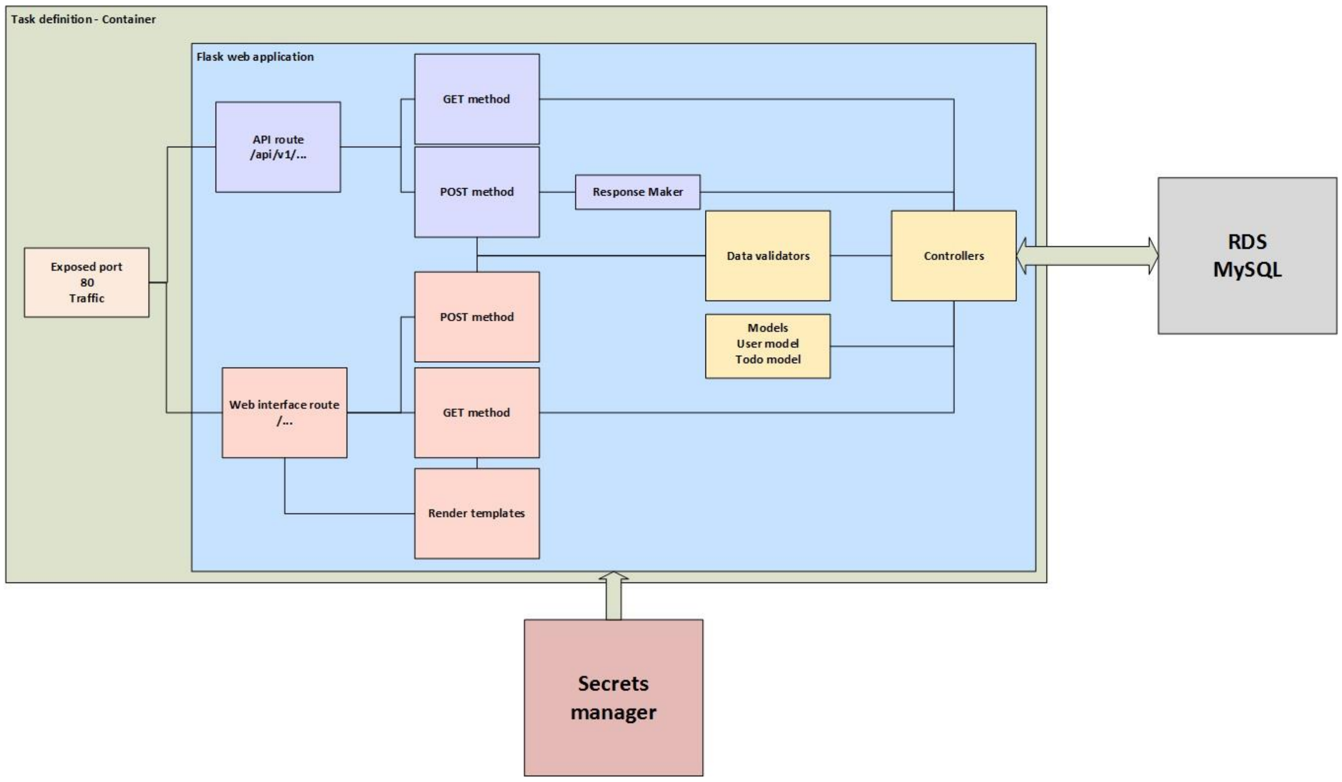
Using mentioned resources above, was created project to complete the task. Project code is push to github.com repository.

3. Web application

Web application created using Python programming language and native Flask framework. Web application have user interface and API. User interface used for browser usage. API will be used in future for mobile applications. Workflow diagram of application:


Security group ECS
Security group Application load balancer
Security group Database
Application load balancer
ECS
Task definition Container
RDS MySQL
Secrets manager

Flask application block diagram:



3.1. Front-end development

Web user interface is used for users to interact with system using browser. Pages user is able to access:

- Index
- Login
- Register

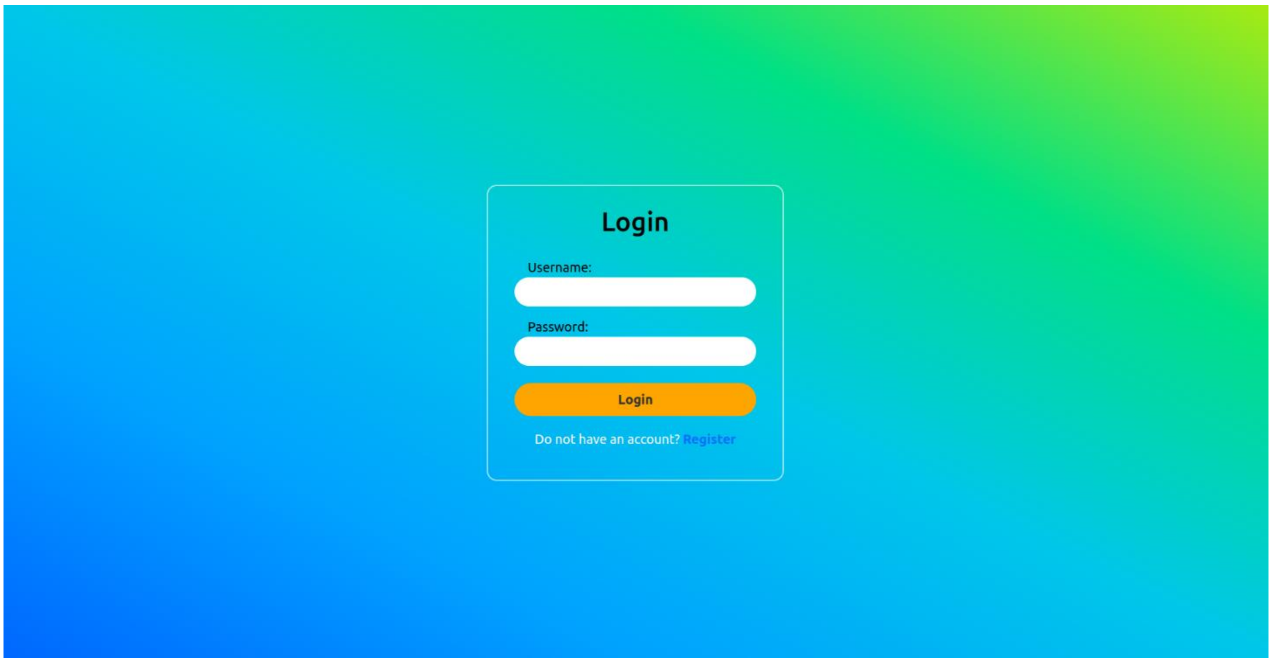
Pages to look similar, created layout.html. This layout used in all pages to not repeat common code. Pages and layout files are located in ***templates*** directory. Style (.css) and javascript (.js) files are located in ***static*** directory.

Unregistered users are able to access login and registration pages. If user is not logged in, system always redirects to login page.

3.1.1. Login page

Login page is a root page. This page is used to let user log in to system. All fields are required. After successful data validation user will be redirect to Index page. If user does not have an account, he can register. Registration page activates, when user registration link is activated.

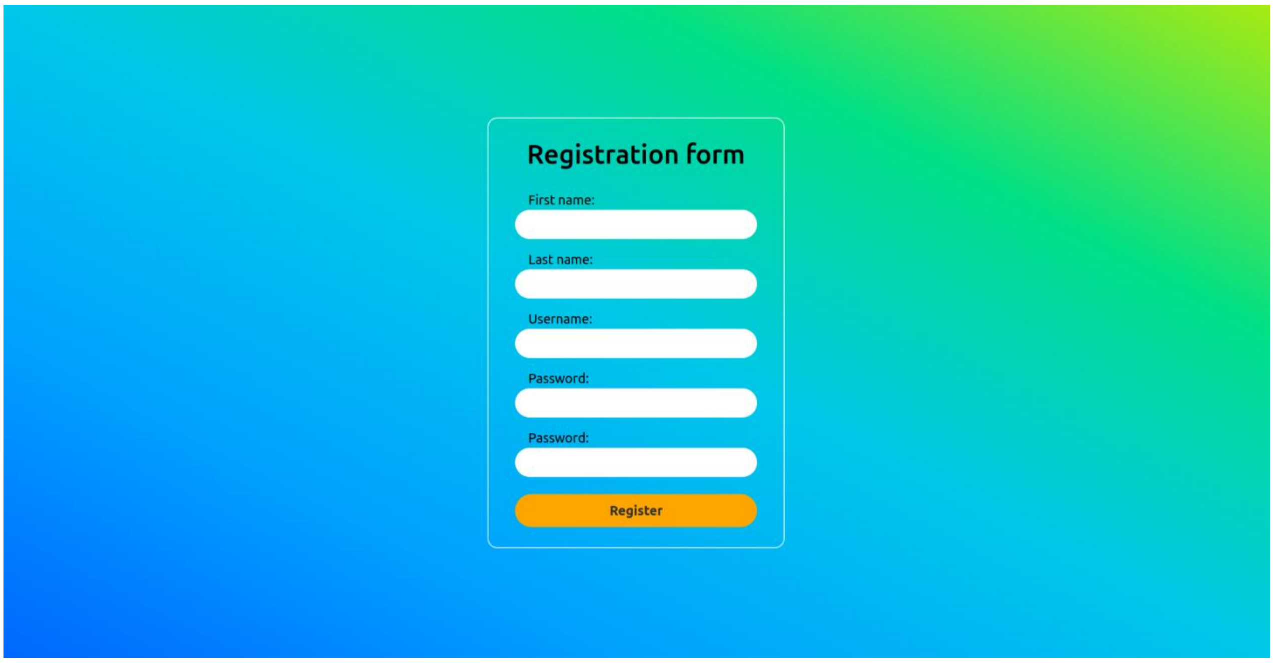
If data is not valid, system will show error messages in top of the page.



3.1.2. Registration page

Registration page lets new user to fill required fields. All fields are required. If given data is valid then user sensitive information is encrypted and saved to database. After successful registration, user will be redirected to login page.

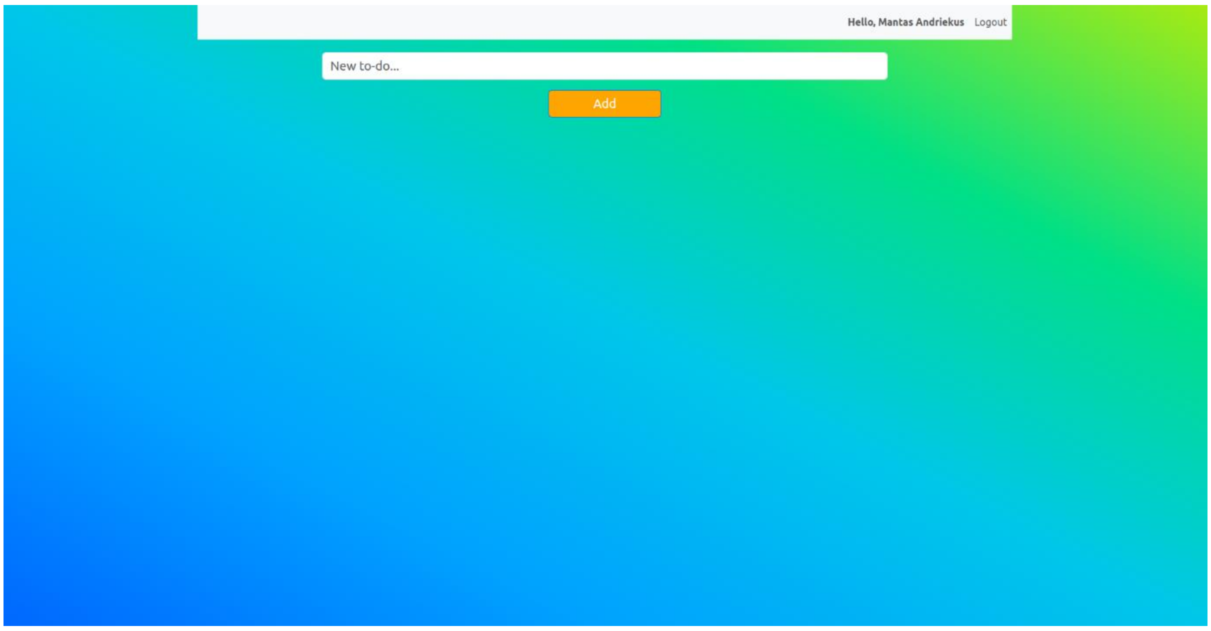
If data is not valid, system will show error messages in top of the page.



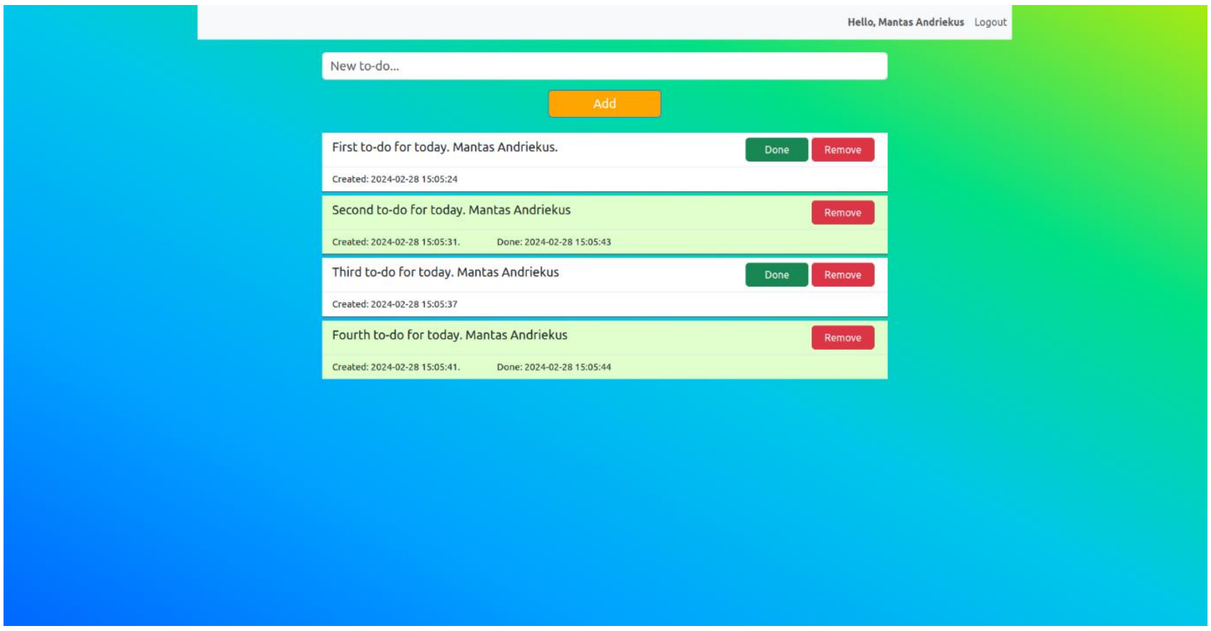
3.1.3. Index page

Index page can access only registered and logged in users. In this page user actions:

- Check to-do list;
- Add new to-do to list;
- Set to-do as done;
- Remove to-do from the list;



Every user registered to the system are able to access only their own to-do list.



First to-do for today. Mantas Andriekus. Done Remove
Created: 2024-02-28 15:05:24

Second to-do for today. Mantas Andriekus Remove
Created: 2024-02-28 15:05:31. Done: 2024-02-28 15:05:43

Third to-do for today. Mantas Andriekus Done Remove
Created: 2024-02-28 15:05:37

Fourth to-do for today. Mantas Andriekus Remove
Created: 2024-02-28 15:05:41. Done: 2024-02-28 15:05:44

3.2. Back-end development